The Pennsylvania State University

The Graduate School

College of Information Sciences and Technology

PERFORMANCE AND LIMITATIONS OF
DISTRIBUTED INFERENCE AND INFORMATION FUSION IN WIRELESS SENSOR NETWORK APPLICATIONS

by

Richard R. Sherry

© 2005 Richard R. Sherry

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2005
The thesis of Richard R. Sherry was reviewed and approved* by the following:

David L. Hall  
Professor of Information Sciences and Technology  
Professor of Electrical Engineering  
Thesis Adviser  
Co-Chair of Committee

John Yen  
University Professor of Information Sciences and Technology  
Professor of Computer Science and Engineering  
Professor in Charge of Graduate Programs in Information Sciences and Technology  
Co-Chair of Committee

C. Lee Giles  
David Reese Professor of Information Sciences and Technology  
Professor of Computer Science and Engineering

Tracy Mullen  
Assistant Professor of Information Sciences and Technology

Lora G. Weiss  
Associate Professor of Acoustics  
Senior Research Associate

*Signatures are on file with the Graduate School.
ABSTRACT

Recent advances in sensor design and miniaturization have provided the opportunity for the creation of large distributed wireless sensor networks. There has been significant progress in combining sensing, processing, data storage and communications capabilities, in network self organization, and in optimizing communication architectures. In contrast to most other network applications, wireless sensor networks face a number of special challenges and constraints resulting from 1) lack of hardwired connections (no external power sources, low communications bandwidths, higher communication error rates), 2) small physical size (small onboard energy supply, small antennas/acoustic transducers, small low energy sensors) and 3) elevated sensor node failure rates.

One of the key remaining challenges is in the area of inference and information fusion (aggregating/filtering/interpreting the sensor data into useful high level knowledge). Many authors have advocated the use of local distributed inference and fusion algorithms such as the iterative message-passing belief propagation algorithms employed on probabilistic graphical models (Bayesian Networks and Markov Random Fields). However, little research has been performed to assess the performance of these algorithms under the special constraints imposed by wireless sensor network applications.

This dissertation reports on a study investigating issues associated with application of these algorithms to realistic wireless sensor networks configurations. This research has produced results delineating the performance and limitations including communications requirements, energy resource requirements and the impacts of different topologies and architectures such as hierarchical/non-hierarchical topologies, centralized or distributed processing, localized (in local node clusters) or full network models, and node cluster size.
# TABLE OF CONTENTS

List of Figures ................................................................................................................... vi
List of Tables ...................................................................................................................... vii

Chapter 1  Introduction, Problem Statement and Background ......................................... 1
1.1 Introduction .................................................................................................................. 1
1.2 Background .................................................................................................................. 3
   1.2.1 Challenges Associated with Sensor Network Design ........................................... 3
   1.2.2 Application Areas for Large Distributed Sensor Networks ............................... 5
   1.2.3 Modeling and Representation of Sensor Networks Using Graphical Models ............................................................................................................ 6
   1.2.4 Difficulties with Solving Large Networks Exactly .............................................. 8
   1.2.5 Belief Propagation Applied to Sensor Networks ................................................ 8
   1.2.6 Advances Beyond Pearl’s Belief Propagation Algorithm .................................... 10
1.3 Presentation of Research Problem ............................................................................. 11
1.4 Significance of this Research Study .......................................................................... 12
1.5 Overview of the Research Approach ....................................................................... 13
1.6 Summary of Research Contributions ...................................................................... 15

Chapter 2  Wireless Sensor Networks ............................................................................ 16
2.1 Sensor Node Components ......................................................................................... 18
   2.1.1 Processing Unit (MCU) .................................................................................... 18
   2.1.2 Communications Devices ............................................................................... 20
   2.1.3 Sensing Modules ............................................................................................. 22
   2.1.4 Energy Supplies ............................................................................................... 23
   2.1.5 Operating Systems and Programming Languages ......................................... 23
2.2 Sensor Network Architectures .................................................................................. 24
   2.2.1 Heterogeneous versus Homogeneous Networks .............................................. 27

Chapter 3  Probabilistic Graphical Models and Inference Algorithms ............................... 29
3.1 Methods for Inference and Information Fusion ....................................................... 29
3.2 Early Work with Bayesian Networks ....................................................................... 30
   3.2.1 Loopy Graphical Models ............................................................................... 32
   3.2.2 Background on Belief Propagation ............................................................... 34
3.3 Detailed Discussion on Probabilistic Graphical Models ......................................... 34
3.4 Inference Algorithms for Probabilistic Graphical Models ..................................... 40
   3.4.1 Message-Passing Belief Propagation Algorithms ......................................... 44
3.5 Variational Approximation Methods ....................................................................... 47
   3.5.1 Mean Field Approximation ............................................................................ 50
   3.5.2 Bethe Approximation ....................................................................................... 52
   3.5.3 General Kikuchi Approximations ................................................................. 53
3.6 Double Loop Iterative Algorithms .......................................................................... 54

Chapter 4  Research Approach and Methods .................................................................. 60
4.1 Simulation Environment ............................................................................................ 60
4.2 Simulation Problem .................................................................................................. 64
4.3 Probabilistic Graphical Model .................................................................................. 66
4.4 Network Topologies and Architectures ..................................................................... 68
LIST OF TABLES

Table 1 Estimated Values of $\alpha$ and $\beta d^n$ for a Variety of Short-Range Radios ............... 21
Table 2 Simulation Parameters ........................................................................................................ 78
Table 3 Communication Requirements by Task for Each Architecture ..................................... 86
Table 4 Communication Latencies by Task for Each Architecture .............................................. 89
Table 5 Communication Breakdown by Task for Fully Distributed Architecture .............. 106
Table 6 Communication Breakdown by Task for Centralized Architecture ......................... 107
Table 7 Communication Breakdown by Task for Locally Distributed Architecture .......... 108
Table 8 Communication Breakdown by Task for Locally Centralized Architecture .... 109
Table 9 Sequential Communications for Each Architecture ....................................................... 110
Chapter 1  Introduction, Problem Statement and Background

1.1  Introduction

Advances in sensor module design and miniaturization provide the opportunity for the creation of large distributed sensor networks. Significant progress in development of inexpensive, small micro-electrical-mechanical sensor (MEMS) modules (aka motes) has been made. Centimeter sized modules are available today and a near term goal seeks to produce millimeter sized sensing modules with individual costs less than $5 to $10. (Nair 2003). These sensing modules are designed to be smart; to combine sensing, processing, data storage and communications capabilities and to sense the physical environment in a variety of modalities; including acoustic, seismic, thermal, and infrared. These sensing modules are designed for rapid deployment in an environment and for wireless communications among; sensor nodes, local processing nodes and to central processing nodes and information users. Additional capabilities are being aggressively pursued including the ability of the distributed sensing modules to self organize into a network and establish communication links and pathways.

Many applications (both military and civilian) for distributed sensor networks have been proposed including: spatial and geographical surveillance and monitoring (disaster recovery, border monitoring), machinery performance and malfunction monitoring and diagnosis (in a residential home, manufacturing plant, on military platforms), battlefield operations (detecting, locating, tracking, and identifying targets), “smart” buildings (residential/commercial), situation awareness (battlefield, residential home, office, manufacturing facility) and context awareness for applications on computational devices (Estrin, Govindan et al. 1999; Chong and Kumar 2003; Sherry and Hall 2003; Zhao, Liu et al. 2003).

The small size of these devices introduces significant constraints on there use. Small batteries and hence limited on-board energy supplies are typical. The small size also
limits the size of communication devices such as radio frequency (RF) antennas and consequently their communications range is limited. On-board computational capabilities including memory and processing speed are also generally limited.

This dissertation describes research that has produced advances in methods and techniques for efficiently performing inference and information fusion in large distributed wireless sensor networks that balances communications and energy costs, communication delays and inference/information fusion quality.

Inference is the process of computing an answer to some query given some input information (evidence) such as a sensor data. Information fusion encompasses the framework, processes and methods for the combination of data and information originating from multiple sources with the objective of obtaining information of higher quality for specific applications.

Sensor networks consist of large numbers of interconnected computational and sensing devices that are distributed in an environment. Sensor networks are designed to provide an information consumer, whether a human or computer application, with useful high-level information concerning some feature, or collection of features, regarding the state of the sensed environment. The features of interest may be discrete entities such as the location, identity and movement of animals, humans or vehicles or the features may represent distributed physical phenomena (a field) such as the temperature, pressure, humidity, or contaminant level across a region of space. The usual goal is to allow the information consumer to intervene in the environment and affect beneficial control over some aspect of the situation. The sensor network may have an a priori task or may be designed with the capability of dynamic tasking and answering emergent query needs. An exciting and potentially very useful application of large distributed multimodal sensor networks is the ability to generate real-time situational (context) awareness of their environment for human participants or for computer applications.
Sensor networks have the potential for greatly increasing the quantity and quality of useful information from the environment and for decreasing the uncertainties associated with this information. Large numbers of sensors afford the opportunity for broad and dense coverage of the sensed environment. Dense sensor placement allows for multiple sensor coverage of the same object/phenomena and hence for uncertainty reduction through intelligent fusion of multiple sensor readings. Furthermore, the dense coverage supports a robust sensing environment which is resilient in the face of sensor failures and noise.

This chapter presents the background for the study, describes the research problems that were investigated, describes the significance of this research, and summarizes the research approach.

1.2 Background

This section presents the context for the research study including the challenges associated with the development and introduction of large distributed wireless sensor networks, examples of the increasing usage of probabilistic graphical models for performing inference and information fusion in complex networks, and recent successes of particular types of inference techniques such as belief propagation in error correcting coding/decoding applications (Turbo-Codes)(Berrou, Glavieux et al. 1993).

1.2.1 Challenges Associated with Sensor Network Design

Rapid advancements have occurred in developing small energy efficient sensor modules and in the self organizing capabilities of network systems. However, significant challenges remain with regard to integrating inference and information fusion into these distributed sensor networks. These challenges include development of algorithms that (Jameson 2001; Kumar, Shepard et al. 2002; Luo, Yih et al. 2002):

- Are flexible for multiple network topologies
- Are robust to nodes failures and sensor errors
• Consider communications constraints (bandwidth limitations) such as those imposed by limited onboard energy supplies, the possible need for stealth, the small size of communication devices such as antennas, location and terrain imposed constraints, and potential countermeasures such as jamming
• Are resilient in the presence of communications errors and noise
• Are scalable with increasing network size (number of sensor nodes)
• Efficiently use network resources (e.g. limited energy supplies)
• Allow for the integration of information from multiple spatially distributed heterogeneous sensors (that are possibly mobile)
• Handle asynchronous communications (due to requirements for energy conservation or stealth), and
• Are adaptable in dynamic uncertain environments (changing/emergent conditions, unpredictability)

There are also challenges related to competing requirements for onboard processing resources for communications, systems control, inference and information fusion, and for decision support and operator interfaces. Issues associated with the data and sensors themselves include the integration of data from heterogeneous sensors, the extent of overlap of sensor coverage, how to handle non-independent sensor data and the sheer number of sensors in the network (Estrin, Govindan et al. 1999).

Real world sensor networks inherently operate in an uncertain environment. This uncertainty results from several factors including: imperfect understanding of the environment; incomplete knowledge of the state of the environment at the time when an action is to be performed; randomness in the mechanisms governing the behavior of the environment; the dynamic and partially observable nature of the environment; and noise, bias and failure in the sensors themselves. Hence, it is never possible to sense the exact state of the environment. Thus, we must reason with this uncertain information to infer coherent beliefs about the state of the environment. Inference and information fusion in real world domains requires the use of methods that can explicitly handle this inherent uncertainty (Luo, Yih et al. 2002).
1.2.2 Application Areas for Large Distributed Sensor Networks

1.2.2.1 Military

For military applications, networks of large numbers of densely distributed, small, low-cost, wireless sensor nodes that can be rapidly deployed (underwater, on the ground or in the air) and which have the ability to self-organize are envisioned. These sensor networks are proposed for applications such as monitoring enemy movements, detecting hazardous chemicals, treaty compliance monitoring and for deployment in hostile environments where it is too dangerous for humans to operate (NRC 2001; Akyildiz, Su et al. 2002; Chong and Kumar 2003).

1.2.2.2 Manufacturing

Industrial applications of distributed sensor networks include inventory control (monitoring the complete life cycle of a product from the manufacturing floor, through shipping and distribution, to the end consumer), industrial process monitoring (monitoring temperatures, pressures, flows), machinery status monitoring (monitoring vibrations, temperatures, humidities, fluid levels and quality, etc. to provide diagnostic information and indications of potential failures) (Akyildiz, Su et al. 2002; Chong and Kumar 2003; Zhang 2004).

1.2.2.3 Building Automation and Monitoring

Distributed sensor networks are proposed for monitoring the entire environment state of building to lower energy usage costs (monitoring power demands and energy losses) and for controlling dust and pollutant concentrations in the air. Sensor networks have also been proposed for monitoring the structural stability of buildings, bridges and other civil structures in regions subjected to high seismic activity (Akyildiz, Su et al. 2002; Chong and Kumar 2003; Whang, Xu et al. 2004; Xu, Rangwala et al. 2004).
Humans have the innate ability to develop and internally construct the context of everyday situations they encounter in the real world. This ability enables humans to communicate more effectively with each other and to generally make appropriate decisions and take effective actions in the pursuit of their objectives and goals. In contrast, computational devices generally have a very limited awareness of the context of the situation in which they are embedded.

To support context-aware computing, networks of diverse sensors are required to determine the identity, location, activities, affective state and the context of the interactions of humans and objects in the environment. Also required is an awareness of environmental conditions (e.g. temperature, day/night, time) and social conditions (presence of other humans). This capability is often referred to as context or situational awareness. Sherry and Hall (2003) discuss application areas for context aware computing and the types of sensors required for developing context awareness in computational devices. Improving the computer’s access to context, will improve the ability of computational devices to communicate more meaningfully with humans, and will facilitate decision-making and the selection of appropriate actions and behaviors.

Other application areas for distributed sensor networks include: intrusion detection, animal habitat monitoring, tracking endangered species, remote patient monitoring, disaster relief, precision agriculture, transportation, and border monitoring (NRC 2001; Akyildiz, Su et al. 2002; Mainwaring, Polastre et al. 2002; Chong and Kumar 2003; Zhang 2004).

This section introduces the problem of representing and performing inference and information fusion in large distributed sensor networks. The question naturally arises why
choose to model sensor networks using probabilistic graphical models? What benefits
does this representation scheme offer that others don’t?

A probabilistic graphical model (PGM) consists of a graphical model and an associated
probability model. The nodes in the graphical model represent random variables. The
edges connecting nodes represent dependencies between the nodes and the lack of an
edge represents an assumption of conditional independence. The parameters of the
probabilistic graphical model are local conditional probabilities or potential functions
relating the nodes. Two general classes of PGMs exist; undirected and directed. For
undirected graphical models (also called Markov Random Fields or Markov networks),
the edges connecting nodes are not directed. For directed graphs, the edges are directed
and the direction of the edge generally represents a causal relationship between the nodes.
Bayesian Networks (a.k.a. Belief Networks) are a special class of directed graphs in
which directed cycles are not allowed. These are known as directed acyclic graphs
(DAGs). PGMs provide a powerful tool for dealing with uncertainty and complexity; two
characteristics of realistic models of sensor networks and other real world phenomena
(Jordan 1998). Special cases of probabilistic graphical models, used in many engineering
and science domains, include mixture models, factor analysis, hidden Markov models,
Kalman filters and Ising models (Jordan 1998; Murphy 2001).

Probabilistic graphical models (PGMs), including Bayesian Networks and Markov
Random Fields, have become one of the dominant uncertainty knowledge representation
and inference technique in artificial intelligence, statistical inference and machine
learning. Many different classes of inference problems have been successfully addressed
using probabilistic graphical models, including medical diagnosis, map learning,
language understanding, fault diagnosis, weather prediction, stock market forecasting,
robot control, real-time monitoring, evaluating data from interplanetary probes and deep
space explorations, information retrieval, image analysis, decision support systems for
antiterrorism risk management, error correcting coding and decoding, consumer help
desks, nuclear reactor diagnosis, credit assessment, computer network diagnosis, and
distributed and remote sensing and sensor fusion (Fung and Del Favero 1995;
Heckerman, Mamdani et al. 1995; Peot 1999; Freeman, Pasztor et al. 2000; Weiss 2000; Hudson, Ware et al. 2001; Yedidia, Freeman et al. 2001; Ware, Beverina et al. 2002; Crick and Pfeffer 2003). A major issue with probabilistic graphical models, however, is that many of the algorithms for their solution for large realistic problems are computationally intractable.

1.2.4 Difficulties with Solving Large Networks Exactly

Why not solve the inference problem and information fusion exactly? Calculating exact solutions for the posterior marginals on nodes in a general Bayesian belief network has been demonstrated to be NP-hard (Cooper 1990). As a result, Cooper indicates “that research should be directed away from the search for a general, efficient probabilistic inference algorithm, and toward the design of efficient special-case, average-case, and approximation algorithms.” Even the calculation of certain approximate solutions (such as stochastic simulation) is, in general, NP-hard (Dagum and Luby 1993). Hence, there is a strong motivation to discover approximate algorithms that yield good approximations to the exact solution and are computationally tractable.

1.2.5 Belief Propagation Applied to Sensor Networks

Pearl (1988) introduced a belief propagation (BP) algorithm (the polytree algorithm) for the solution of Bayesian Networks which are singly-connected (there are no closed loops in the network – that is, there is only a single path between any two nodes in the network). BP is a decentralized iterative algorithm that operates by transmitting messages between nearby nodes (random variables-RV) in the probabilistic graphical model. Each node in the graphical model acts as a processor which can communicate only with its nearest neighbors. For a Bayesian Network the nearest neighbors correspond to the node’s parents and children. The processor associated with each node maintains knowledge of the conditional probabilities of its states conditioned on its parents’ states. When a node processor is activated it reads all incoming messages from its parents and children, updates its belief, and sends out new messages to it parents and children. Each node computes a belief regarding the state of the RV associated with the node by
combining messages it has received from its neighboring nodes. The messages contain all the information (evidence) necessary to locally evaluate the posterior marginals for each variable. BP is an exact algorithm for solution of singly-connected networks. As discussed in Chapter 2, it has been recently discovered that it is a surprisingly good approximate algorithm for more general networks containing loops.

Belief propagation is an ideal approach for computations and communicating beliefs in sensor networks because of its compact representation, its distributed nature and its robustness in the presence of noise and network degradation (Crick and Pfeffer 2003). Crick & Pfeffer (2003) have performed empirical simulations that suggest that Pearl’s belief propagation algorithm is surprisingly robust for simulated sensor networks under realistic conditions where the environment is changing rapidly (the environment state is changing at a rate commensurate with the time for beliefs to propagate in the network). Furthermore, their results indicate that BP is quite robust to highly asynchronous communications and failure of network nodes. They also observed that the effects of the node failures are localized to a region near the failed node.

Chen, Wainwright et al. (2003) have applied a variant of the max-product algorithm (which is the “belief revision” form of the belief propagation algorithm) to the multitarget-multisensor data association problem. This problem involves developing the most probable association of sensor measurements and target tracks. They note that this problem has high computational complexity for high target densities and large numbers of sensors where the number of possible associations between sensor measurement and target track increases combinatorially. Their algorithm will either converge, or will indicate failure to find the maximum \( a \text{ posteriori} \) probability (MAP) estimate. They ran tests of their algorithm on simulated sensor networks with various target densities and measurement noise. In addition, for some tests they also simulated missed detections and false detections. In all test cases their algorithm converged to the unique solution within approximately ten iterations.
1.2.6 Advances Beyond Pearl’s Belief Propagation Algorithm

Relatively recently a class of inference methods have been introduced which are related to belief propagation and reformulate the belief propagation inference problem as an energy minimization problem. It has been shown that there is a relationship between the belief propagation algorithm and certain approximations (Bethe 1935; Kikuchi 1951) to the variational free energy (Gibb’s free energy) in statistical physics (Yedidia, Freeman et al. 2000; Aji and McEliece 2001). Yedidia et al. show that BP can only converge to a stationary point of the Bethe free energy and they have found that belief propagation corresponds to iterative descent down an associated Bethe free energy. In (Yedidia, Freeman et al. 2001) they introduce a class of generalized belief propagation (GBP) algorithms that minimize an arbitrary Kikuchi free energy which were demonstrated to outperform BP algorithms (either fewer convergence cycles or GBP converged when ordinary BP did not). Yuille (2002) introduced a class of provably convergent iterative algorithms that exploit the structure of the approximate free energy function (Bethe or Kikuchi) and are alternatives to BP and generalized belief propagation. Their algorithm, called the Concave Convex Procedure (CCCP), operates by decomposing the free energy function into concave and convex parts, and introduces a double loop iteration scheme where the inner loop determines the constraints on the beliefs (see chapter 3) and the outer loop iterates the update rules (subject to the constraints) for the beliefs such that the free energy is reduced during each iteration. Heskes, Albers et al. (2003) also developed a class of double loop algorithms, in which the inner loop corresponds to constrained minimization of a convex bound on the Kikuchi free energy and the outer iteration loop recalculates the bound. Teh and Welling (2001) introduced the Unified Propagation and Scaling Algorithm (UPS) which shares certain similarities with the Heskes, Albers et al. algorithm. However, instead of bounding the portions of the concave entropy, UPS clamps some of them (clamps a number of hidden nodes to their current marginals) during the iteration.
1.3 Presentation of Research Problem

Straight forward application of existing algorithms such as Joint Probabilistic Data Association (JPDA) and Multiple Hypothesis Tracking (MHT) to the multitarget-multisensor (MTMS) tracking and data association problem for large sensor networks suffer from a combinatorial problem in possible data associations (Zhao and Guibas 2004). The complexity grows exponentially with the number of sensors. Approximate solution techniques such as Pearl’s (1988) belief propagation algorithm applied to graphical models of realistic sensor networks have been shown to provide solutions that are good approximations to the exact solution for sensor network inference problems in general (Crick and Pfeffer 2003) and for the data association problem specifically (Chen, Wainwright et al. 2003). Furthermore, the complexity of belief propagation type approximate algorithms is approximately linear in the number of sensor nodes (Chen, Wainwright et al. 2003).

With respect to application of these approximate solution techniques to large wireless sensor networks, a number of issues are under-researched including the performance of the algorithms for different network architectures. The principal performance metrics are the quality (accuracy, precision, error rates, etc.) of the inference and information fusion results, the communications and energy costs, and the communications delays. Research is needed in these areas for developing guidance for the integrated design of distributed wireless sensor networks, the associated network models, and inference and information fusion algorithms.

The specific research problems that are addressed in this research are:

- Demonstration of the application of PGMs and approximation solution methods (ASMs) to realistic sensor networks and understanding their performance
- Development of methods and techniques for localization of the inference and information fusion processing that balance the competing requirements for solution quality, communications and energy resources usage and communications latencies
• Production of results delineating the performance and limitations of PGMs and ASMs in terms of these metrics for four diverse sensor network processing architectures:
  o Centralized processing
  o Fully distributed processing
  o Locally centralized
  o Locally distributed

• Development of guidelines for the application of PGMs and ASMs

1.4 Significance of this Research Study

This section briefly describes the expected significance and impacts of wireless sensor networks and the importance of research on efficient methods for performing inference and information fusion in wireless sensor networks. Chapter 6 discusses these topics in greater depth.

In the February 2003 issue of MIT's magazine of innovation, Technology Review, Wireless Sensor Networks were identified as one of the ten emerging technologies that will change the world. A market research firm in San Diego, ON World, which specializes in wireless networking, estimated that wireless sensor networks sales will top $7 billion by 2010 (OnWorld 2004). DARPAs Information Technology Office states “Networked micro sensors technology is a key technology for the future” (SensIT).

Results that provide guidance for, and facilitate the use of advanced approximation methods for efficient solution of inference problems (including Bayesian Networks) have the potential for a dramatic impact in many domains of application. An outstanding example of the potential impact is the introduction of the Turbo Codes algorithm in the area of error-correcting coding in communications. The use of Turbo Codes (an application of loopy belief propagation) has been estimated to improve the performance of error correction coding/decoding from several decibels to within about 1/2 decibel of the theoretical Shannon limit for the error rate (Zhu 1999). Zhu indicates that a 1 decibel
reduction in the error rate can translate into $80 million dollars in savings for satellite transmissions.

MacKay et al. (1998) have noted with respect to broader applications of these algorithms; “We believe that there are general undiscovered theorems about the performance of belief propagation algorithms on loopy DAGs. These theorems, which may have nothing directly to do with coding or decoding, will show that in some sense BP “converges with high probability to a near optimal value” of the desired belief on a class of loopy DAGs … If such theorems exist, they will no doubt find applications in realms far beyond information theory”.

Solution techniques for the message passing algorithms such as belief propagation (BP) and many of the advanced approximation methods that extend BP are considered ideally suited for performing inference in distributed applications since processing can be performed locally at a node and in parallel with processing performed at other nodes. The distributed nature of these algorithms provides a robustness against node failures and can accommodate changes to the network topology (e.g. mobile nodes entering or leaving a specific locality) without reconfiguration of the entire network. However, as Meguerdichian, Slijepcevic et al. (2001) indicate, the development of practical, localized algorithms is probably the most needed and most challenging task in wireless ad-hoc sensor networks.

1.5 Overview of the Research Approach

This section presents a brief overview of how the research was performed and the research methodology, what other research techniques/methods were considered, and why this approach is appropriate to address the research problem. Chapter 4 presents an in-depth discussion of the research methodology and techniques.

The approach that was taken to address the research problem was to construct simulated sensor networks with various architectural configurations for the graphical models and to
perform a matrix of simulation runs that test the performance of the approximate belief propagation algorithms. The information generated from these simulation runs was then used for developing guidance for the integrated design of distributed wireless sensor networks, the associated network models, and inference and information fusion algorithms.

Chapter 4 discusses how simulated sensor networks topologies and characteristics were determined and what types of graphical models (e.g., Bayesian Belief networks, Markov Random Field) were utilized in modeling these simulated sensor networks. Chapter 4 also discusses the various types of approximation algorithms that were implemented in the research and why these types were selected. The principal test parameters (independent variables such as network topologies, localizing and clustering techniques, cluster size, model simplification techniques, etc.) and the principal outcomes of interest (performance metrics) are also identified.

The canonical problem to be used for examination of the performance of advanced approximation algorithms in distributed sensor networks is the data association problem for the multitarget-multisensor (MTMS) localization and tracking. This problem was selected for a number of reasons. First, localization and tracking is a required capability in many sensor network applications. Second, the data association problem exhibits many of the central challenges to distributed inference and information fusion, including how to dynamically configure collaborative sensor node groups, and how to perform collaborative distributed processing, information sharing and sensor tasking and control. Third, the localization and tracking problem spans a broad spectrum of complexity; from locating a single stationary target through locating and tracking multiple mobile targets with multiple sensors.

Chapter 4 also discusses the sensor network simulation environment in which to test algorithm behavior. Simulation environments that are considered range from the most realistic environment that uses actual sensor network physical components (sensor modules) and software to construct a real sensor network, to the simulation of sensor
networks at a high level of abstraction on a PC/workstation using a common OS
(Windows XP/Linux) and high level language such as C/C++ or in a specialized
environment such as Matlab. In between these two extremes are approaches involving
using simulation software that allows low level simulation of the actual sensor node
hardware and software and uses the actual software that runs on the physical sensor
network.

1.6 Summary of Research Contributions

The major contributions of this research are the development of inference and
information fusion techniques that are scalable to large sensor networks, that can be
applied locally and that minimize the consumption of scarce network resources (energy
supplies and communication bandwidth).

The specific contributions include:

- Development of techniques for dynamic generation of graphical models
- Procedures for localizing the inference and information fusion computations
  (clustering)
- Methods for model simplification to address cross-cluster dependencies
- Identification and quantification of important performance metrics
  - Inference quality
  - Communications costs (energy usage)
  - Communications latencies
- Quantitative results for the performance metrics for four diverse network
  architectures

Chapter 6 discusses these contributions in greater detail.
Chapter 2 Wireless Sensor Networks

Wireless sensor networks consist of many small modules (nodes) distributed in the environment. These devices consist of four main components; a processing unit (microcontroller unit-MCU and data/program memory), communication devices such as a radio transmitter/receiver (transceiver), an onboard energy supply (typically a battery) and various sensing submodules.

Sensor node modules currently being developed span a considerable range in size and capability. Small sensor node modules such as those developed in the UC Berkeley Smart Dust program are centimeter sized (Figure 1). These small sensor node modules are often referred to as motes. Commercial versions of these small sensor node modules, such as the MICA2 mote offered by Crossbow Technologies are equipped with the 16 MHz 8-bit Atmel ATmega128L processor with 8K bytes SRAM, 128K bytes of program flash memory and 512K bytes of measurement (serial) memory. On the other end of the spectrum are higher capability sensor node modules often referred to as gateways, such as the Crossbow Stargate module powered by the Intel Xscale PXA255 processor. The Xscale processor is the same processor found in a number of high end handheld computers (PDAs). The Intel Xscale PXA255 processor is a 32 bit 400 MHz RISC processor with 64 MB SDRAM and 32 MB flash memory.

There is a large difference in the computational capabilities across the range of sensor node modules. The processors in small motes can process four million to 10 million instructions a second, whereas the processor in a PDA can perform about 400 million a second (Culler and Mulder 2004).

One of the principal goals of sensor network node design is to develop sensor nodes that are relatively inexpensive such that they can be deployed in large numbers to afford dense coverage of the sensed environment and to allow continued network operation should some fraction of the nodes fail. Current small-scale motes such as those
developed by Intel Research and UC Berkeley have recently become commercially available at $50-$100 each. However, it is expected that motes will drop in price to less than $5 each over the next five years (Intel 2004).

A number of limitations constrain the design and operation of wireless sensor networks. These include few or no hardwired connections. Hence, communications bandwidth is controlled by the wireless communications capabilities. Furthermore, the energy resources available for processing, sensing and communications are limited to onboard storage (battery) capacity. The small size of the sensor nodes also translates into small antenna size for radio transceivers and the attendant range and power limitations.

Because of limitations on energy resources and communications bandwidth, it is frequently necessary in the design and implementation of wireless sensor networks to tradeoff inference quality for energy (Min and Chandrakasan 2003). In wireless sensor networks it is advantageous to process (aggregate/fuse) the data as near the data source as
possible in order to reduce the amount of information that must be communicated. Hence, trade-offs between data processing and wireless communication are possible. Local data processing can be a critical consideration in minimizing power consumption in a wireless sensor network (Salhieh and Schwiebert 2004).

2.1 Sensor Node Components

2.1.1 Processing Unit (MCU)

The processing unit provides control of the sensors, control and execution of communication protocols, and executes the data processing, inference and data fusion/aggregation algorithms on the collected sensor data. Examples of high end and low end processing units include:

- High End - Intel StrongARM/Intel Xscale PXA255 Processor (Figure 2)
- Low End – Smart Dust (Atmel AVR microcontroller)

The energy consumption of these MCU units range from 400mW of power while executing instructions for the Intel StrongARM processor (an earlier Intel design to the Xscale processor) (Raghunathan, Schurgers et al. 2002) to about 16.5 mW for the Atmel ATmega103L AVR microcontroller (Raghunathan, Schurgers et al. 2002). Figure 3 presents the power requirements for a range of processors spanning those found in small sensor nodes (motes) through desktop and server computers.

For managing power, MCUs typically support different operating modes including Active, Idle, and Sleep modes. Each mode has a different power requirement. The Intel StrongARM MCU, for example, consumes 50mW of power in the Idle mode and 0.16mW in the Sleep mode (Raghunathan, Schurgers et al. 2002). Transitioning between operating modes generally imposes additional power requirements and has associated time delays (Raghunathan, Schurgers et al. 2002).
Figure 2 Stargate Processing Platform

Figure 3 Power Requirements of Various Computational Devices
(Source: http://www.intel.com/research/exploratory/motes.htm)
2.1.2 Communications Devices

The most common communication devices for wireless sensor networks are radio transceivers and acoustic transmitter/receivers for underwater applications.

Radios generally can operate in four distinct modes, Transmit, Receive, Idle, and Sleep (Raghunathan, Schurgers et al. 2002). For most radios, power consumption in the Idle mode consumes nearly as much power as in the Receive mode (Xu, Heidemann et al. 2001). Consequently, to maximize energy consumption, it is necessary to completely shutdown the radio (Sleep mode) while not transmitting or receiving messages.

As for MCU, there is significant power consumption cost for transitioning between radio operating modes. This is particularly true when transitioning from Sleep mode to Transmit mode resulting from energy requirement during transmitter startup (Wang, Cho et al. 2001).

A power law of the form shown in equation (2.1) is often used to describe the radiated power loss when transmitting over a distance $d$ and with path loss exponential term $n$.

$$\frac{P_r}{P_i} = \beta d^n$$  \hspace{1cm} (2.1)

The value of the exponential term $n$ for RF communications is generally taken to be in the range 2 to 4, where a value of 2 corresponds to free-space falloff. For small centimeter-sized, ground-level radio transmitters/receivers, Pottie and Kaiser (2000) and Merrill, Liu et al. (2004) indicate that the value of $n$ is approximately 4 as a result of attenuation, mainly due to destructive inference from ground reflected signals.

To reduce the power requirements associated with long distance radio transmissions, it is often beneficial to hop the communication from node to node rather than transmit directly to a distant receiving node. For multi-hopped communications equation (2.1) can be modified as shown in equation (2.2)
\[ \frac{P_r}{P_t} = h \beta \left( \frac{d}{h} \right)^n \]  

(2.2)

where \( h \) is the total number of hops to equally spaced nodes.

A value of 4 for the power loss exponential term \( n \) as suggested by Pottie and Kaiser results in the power transmission requirements increasing by a factor of 16 for each doubling of the communications distance for single hop communications. However, if the additional distance is covered by multi-hopping (doubling the number of hops) then the communication costs only increase by a factor of 2. With a \( d^4 \) energy requirement, sending a bit over 10 or 100 meters consumes as much energy as thousands/millions of processing operations (Pottie and Kaiser 2000; Culler and Mulder 2004).

There are other fixed costs associated with transmitting and receiving communications which may impact the apparent energy advantage of multi-hopping communications. Min and Chandrakasan (2003) note that equations (2.1) and (2.2) fail to account for all power requirements for transmitting and receiving messages. They suggest a “more complete model for energy consumption per bit is, \( \alpha + \beta d^\alpha \) where \( \alpha \) is a distance independent term that accounts for the real-world overheads of transmitter and receiver electronics … and digital processing.” As shown in Table 1, “for short-range radios, \( \alpha \) can substantially exceed the maximum value of the \( \beta d^\alpha \) term!”

<table>
<thead>
<tr>
<th>Radio</th>
<th>Data Rate</th>
<th>( \alpha )</th>
<th>Max. ( \beta d^\alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFM TR1000</td>
<td>2.4 Kbps</td>
<td>14 ( \mu )J</td>
<td>3.1 ( \mu )J</td>
</tr>
<tr>
<td>RFM TR1000</td>
<td>115.2 Kbps</td>
<td>372 nJ</td>
<td>65 nJ</td>
</tr>
<tr>
<td>AMPS-1</td>
<td>1 Mbps</td>
<td>570 nJ</td>
<td>740 nJ</td>
</tr>
<tr>
<td>Cisco Aironet 350</td>
<td>11 Mbps</td>
<td>236 nJ</td>
<td>91 nJ</td>
</tr>
<tr>
<td>Atheros ISSCC02</td>
<td>54 Mbps</td>
<td>14.8 nJ</td>
<td>11 nJ</td>
</tr>
</tbody>
</table>

Table 1 Estimated Values of \( \alpha \) and \( \beta d^\alpha \) for a Variety of Short-Range Radios

(Min and Chandrakasan 2003)
Min and Chandrakasan (2003) suggest “To avoid surprises, energy-efficient communication protocols must be designed around accurate energy models of the targeted hardware.”

2.1.3 Sensing Modules

Sensor modules measure physical signals (phenomena/energy) in the environment and translate these physical signals into electrical analog or digital signals. The types of physical signals that can be sensed include:
- Infra-red
- Visible light/images
- Temperature
- Humidity
- Acoustic
- Vibration
- Seismic
- Motion
- Strain
- Acceleration
- Magnetic field intensity

Sensor power consumption includes signal reception and sampling, conversion of physical energy to electrical signals, conditioning and processing the signals, and analog to digital conversion (Raghunathan, Schurgers et al. 2002). Passive sensors (sensor devices that detect the energy emitted by environmental objects without emitting any energy signals themselves) such as acoustic, temperature, seismic etc., consume relatively small amounts of power compared to other sensor module components. On the other hand, active sensors such as radar and active sonar, imagers, and pan-zoom-tilt cameras can require significant power (Raghunathan, Schurgers et al. 2002).
2.1.4 Energy Supplies

Energy consumption is the single most critical factor determining the lifetime of individual sensor nodes and the effective lifetime of the overall sensor network. The predominant power source for sensor nodes are on-board batteries. For many applications, the required operating lifetime may be weeks, months or even years and battery replacement may not be a feasible option for large distributed sensor networks embedded in remote or hazardous environments (such as battlefields). Other types of energy sources are being investigated that involve extracting energy from the local environment. Unfortunately, these sources of energy are often intermittent and may only provide limited power. The energy sources that have been mentioned for powering sensor node modules include:

- Battery – A typical AA size alkaline battery (used in the Crossbow MICA2 mote) stores about 3 watt-hours of energy (zbattery 2005) as do standard coin sized batteries (Culler and Mulder 2004)
- Photovoltaic – A one centimeter square solar cell produces about 10 milliwatts of power (Culler and Mulder 2004)
- Motion/vibration (Paradiso and Starner 2005)
- Thermoelectric energy conversion (Paradiso and Starner 2005)

2.1.5 Operating Systems and Programming Languages

TinyOS (Hill, Szewczyk et al. 2000) is a node level operating system (OS) developed at UC Berkeley for sensor network applications that run on resource constrained hardware devices such as the Berkeley motes. To minimize the memory requirements for the OS, only those portions of the OS that are required for a platform and for an application are compiled (along with the application software) and stored in mote program memory. The programming language for TinyOS is nesC (Gay, Levis et al. 2003), a dialect of C that is both restricted and extended.
Higher end sensor modules, such as the Crossbow Stargate, typically run a higher level operating system such as Linux for embedded systems.

2.2 Sensor Network Architectures

Liggins II, Chong et al. (1997) have noted that the traditional architecture for data and information fusion is centralized. Valet, Mauris et al. (2001) found, in a recent survey of the information fusion literature (1997-1999), that 89% of the citations were for fusion for centralized fusion architectures and only 11% for distributed fusion architectures. Estrin, Govindan et al. (1999) suggest that localized algorithms in which local computations collectively achieve a desired global objectives may be necessary for distributed sensor network coordination and that since the sensing is distributed it is natural to look at distributed architectures.

In centralized architectures, data from multiple sensors are transmitted to a central location for processing, inference and fusion. At the other extreme are fully distributed architectures where nodes can communicate with other nodes, the fusion and inference process is distributed among all the nodes, and there is no fixed superior/subordinate relationship between nodes (Liggins II, Chong et al. 1997). In between these two extremes lie hierarchical architectures where the network nodes are organized hierarchically from the lowest sensor level nodes, through higher level inference, fusion, and communication nodes. Figure 4 provides examples of these three types of architectures.

A hierarchical architecture in which local sensor nodes are aggregated into clusters has been advocated by a number of authors in order to reduce communications bandwidth requirements and power consumption (Van Dyck and Miller 2001). These clusters could be self-organizing based on proximity to each other, proximity to a known target, or other requirements. Local communications among nodes in the cluster would be short-range,
possibly one hop, and would be mainly for the purpose of distributed detection, inference and information fusion. The local nodes could either communicate their measurements to a master node (cluster head), which would aggregate these beliefs and transmit the processed information to a higher node in the hierarchy or the inference and fusion could be distributed among the nodes in the cluster. In either case, to assure survivability and network robustness against node failures, the local cluster should have the ability to reorganize. This could be achieved, for example, by promoting another node to master node, should the current master node fail.

There are advantages and disadvantages of each sensor network architecture. Liggins II, Chong et al. (1997) indicate that a centralized architecture is theoretically optimal and conceptually simpler than other architectures. However, with very large sensor networks the communication bandwidth requirements and processing requirements on the central node become overwhelming (Liggins II, Chong et al. 1997). Other limitations of centralized architectures that have been identified include a single point of failure in
system, a limited ability of lower level units to operate independently from the central unit, and latency on development of the fused picture at lower level nodes. (Estrin, Govindan et al. 1999; Jameson 2001)

The following benefits and limitations of distributed hierarchical architectures have been identified (Liggins II, Chong et al. 1997; Jameson 2001):

Benefits:

- All nodes have access to the fused picture
- The flow of information back to lower level units is more rapid
- The processing burden at higher level nodes is reduced since processing is distributed among multiple nodes in the network
- Communication bandwidth requirements are generally less than for a centralized architecture since low level sensor data does not need to be sent to the central processing site
- There is no need to maintain a large centralized database since each node has its own database
- The network is more robust since there is not a single point of failure for the network

Limitations:

- Loss of intermediate nodes separates fusion in subsets of network
- There is still a significant processing burden
- Distributed systems are subject to estimation and inference bias from multiple non-independent measurements

When analyzing sensor network architectures, it is useful to recognize that there are at least four superimposed sub-architectures which are interdependent; (1) the physical sensor/network architecture, (2) the communications infrastructure, (3) the inference and data fusion information architecture, and (4) the knowledge/abstraction architecture. The physical sensor/network node architecture defines such properties as the density of sensor and node placement and the grid structure (regular or random placement). The second sub-architecture deals with how communications in the network are organized. Do nodes
communicate directly with a central node, a higher level node in a hierarchical organization or node to node at the lowest levels? Is communications single-hop, multi-hop or somewhere in between? The third sub-architecture involves the distribution and partitioning of inferencing and data fusion. Are these processes fully distributed among all nodes, centralized in one node, or distributed in a hierarchical organization? Finally, if the inference and fusion processing are distributed, what is the abstraction level sub-architecture? At what level of representation is the fusion process performed and where in the network is each level of fusion performed? For example, is direct fusion of low level sensor data performed or are representative features extracted from the sensor data to form feature vectors with subsequent fusion of the feature vectors?

These considerations argue for the integration of communications, and inference and fusion algorithms, during network design and assessment. The interdisciplinary merging of network communications, computation, signal processing, distributed and fault tolerant algorithms, adaptive systems, sensor and information fusion, and decision theory for distributed micro-sensor networks has been termed Collaborative Signal and Information Processing by the signal processing community (Kumar, Shepard et al. 2002). They note “To meet the energy constraints and effectively support end-to-end applications, an ad hoc sensor network must optimize processing and communication among signal processing, data fusion, querying, and routing tasks. For example, diffusion of data through the network may be directed by considering both information gain from signal processing applications and resource usage, thus breaking the traditional abstraction barrier between signal processing and network routing.”

2.2.1 Heterogeneous versus Homogeneous Networks

A homogeneous sensor network consists of sensor node nodules which are all similar in design and capability, whereas heterogeneous sensor networks are composed of dissimilar sensor node modules. Although the heterogeneity of the network may result from differing sensing modalities on the sensor node modules with similar processing and communications capabilities, we are more interested in the case where the heterogeneity
arises from differing processing and communications capabilities among the sensor nodes. A type of heterogeneous network that is being seriously investigated is comprised of a limited number of high capability nodes (such as the Stargate module) interspersed among many small mote type sensor modules (Intel 2005). The higher capability nodes form local communications gateways into the sensor network and facilitate the local processing of inference and data/fusion algorithms that may not be capable of being run on the mote nodes.
Chapter 3 Probabilistic Graphical Models and Inference

Algorithms

This chapter discusses prior work regarding methods for performing inference and information fusion in probabilistic graphical models of large distributed sensor networks.

3.1 Methods for Inference and Information Fusion

Nixon (2003) provides the results of a wide ranging survey of applied inference methods for a broad range of applications. He indicates that these methods can be broadly classified into three classes; logical, connectionist and statistical. Logical methods are largely based on propositional or first order predicate logic and include rule-based expert systems and production systems. Artificial neural networks are the principal exemplar of connectionist methods. Statistical methods include probabilistic, graphical and decision-theoretic based methods. Each class of methods has its particular strengths and weaknesses. Logic based methods such as production systems are particularly good at representing and reasoning about complex phenomena. However, these systems tend to be “brittle” when exposed to uncertainty and contexts outside their knowledge base. Connectionist methods, such as neural networks, are robust in the presence of uncertainty and have well developed learning algorithms. They, however, require large amounts of training data (for real world complex problems). Furthermore, it is difficult to generalize the results of a trained neural network (for example, development of a set of rules that summarize the performance of the network). Probabilistic graphical methods such as Bayesian belief networks and Markov Random Fields are also robust in the presence of uncertainty. They can also learn the values of the parameters of the network and even the structure of the model can be learned. Unlike neural networks (in which the knowledge is contained implicitly within the network structure and weights), probabilistic graphical models are generally transparent and their behavior can be explained and generalized.
3.2 Early Work with Bayesian Networks

Bayesian Networks were introduced during the mid-1980s, largely through the work of Judea Pearl (Pearl 1988). Bayesian Networks (BNs) are directed acyclic graphs (DAGs) where the nodes are random variables, and edges are directed links representing the conditional dependencies between the node random variables (Pearl 1988; Heckerman, Mamdani et al. 1995; Jensen 1996; Jensen 2001). The random variables (RVs) are used to represent events or objects in the real world. Various instantiations of these RVs represent possible configurations for the current state of the world. The relationships between the RVs are described by the edges connecting the nodes.

Probabilistic graphical models (Bayesian Networks and Markov Random Fields) are a merging of probability theory and graph theory. They represent powerful tools for dealing with the two key problems, uncertainty and complexity, that are inherent when developing real world applications.

The random variables associated with the nodes can either be discrete or continuously distributed. The dependencies between nodes in a BN are contained in conditional probability distributions that represent the dependencies of a node on its parents. A BN can be thought of as a knowledge base where the knowledge is represented by the topology of the network and the conditional probability distributions. The BN represents a probability distribution over the set of random variables represented by the nodes in the network.

Bayesian Networks were initially developed primarily to add the capability of using probabilities for representing uncertainty into expert systems (Pearl 1988; Neapolitan 1990). Examples of early expert systems based on Bayesian Networks include: MUNIN: Neuromuscular diagnosis and interpretation of electromyographic findings (Andreassen, Woldbye et al. 1987), Pathfinder: Lymph-node pathology diagnosis (Heckerman, Horvitz et al. 1992) and QMR-DT: a decision-theoretic reformulation of the Quick Medical Reference (QMR) knowledge base (Shwe, Middleton et al. 1991).
Another interesting application of probabilistic graphical models is the Vista system (Horvitz, Srinivas et al. 1992), developed to reduce the cognitive load of human operators (ground controllers) that are responsible for monitoring the propulsion systems of the NASA Space Shuttle. The Vista system integrates information display strategies with a probabilistic reasoning module to perform diagnosis of propulsion systems faults in real time. At the heart of the Vista system is a probabilistic causal network (a Bayesian belief network) constructed by a shuttle propulsion system expert to model the causal relationships and uncertainties among components of the system. The system interprets real time telemetry (sensor data such as oxygen and fuel tank pressures and temperatures, fluid flow rates, and various valve positions) and provides advice on the likelihood and criticality of alternative failures mechanisms. The Bayesian Network developed for the Vista systems contains nodes that explicitly represent component failures including sensor failures.

Probably the most widely used Bayesian Networks applications are the ones embedded in Microsoft productivity application software, including the Office 95 Answer Wizard (Heckerman and Horvitz 1998), the Office 97 Office Assistant, and over 70 Technical Support Troubleshooters (Hedberg 1998). The Microsoft Office 97 Office Assistant (the much loved/hated paperclip guy) was based on work performed in the Lumiere Project (Horvitz, Breese et al. 1998). This project focused on the use of Bayesian Networks and influence diagrams to make inferences about the goals of software users from their observed actions and queries, and from the context of their interactions (the application program state) and then to provide assistance to the software users.

The graphical nature of Bayesian Networks and the causal semantics facilitates a human reasoning-oriented approach to modeling (Heckerman, Mamdani et al. 1995). Heckerman (2002) provides the following benefits accruing from the graphical structure of Bayesian Networks:

- Intuitive interface for modeling
- Modular: Useful tool for managing complexity
- Useful data structure for applying Bayes rule efficiently
- Common formalism for many models
  - Facilitates transfer of ideas between communities
  - Facilitates design of new systems

Heckerman further indicates that Bayesian Networks:
- Encodes the right amount of modularity (“conditional independence”)
- Facilitates understanding of the assumptions
- Facilitates explanation of results / recommendations
- Incorporates causal knowledge

Perhaps the strongest argument for the use of Bayesian Networks is that “they are a direct representation of the world, not of the reasoning process” (Pearl and Russell 2003). The directed edges in the Bayesian Network graph represent real causal relationships between the parent nodes and their children and are not just indicative of the flow of information during the inference process. Bayesian Networks also allow bidirectional inference – given a cause what are the effects, or given an effect what are the likely causes.

3.2.1 Loopy Graphical Models

The complexity of solving a graphical model is not only determined by the number of nodes in the model but also by the structure (topology) of the graph. Graphs which are singly-connected are relatively straight-forward and algorithms that provide exact solutions exist that run in polynomial time. However, if loops exist in the graphical model then the complexity increases dramatically. A loop is an undirected path in a graph that both starts and ends at the same node. If loops exist in the graph then the nodes in the graph are not singly-connected.

Loops in Bayesian Networks are undirected cycles in the underlying network (i.e., neglecting the direction of the link arrows). A network with loops is no longer singly-connected and local message-passing techniques such as belief propagation may not converge. For loopy graphs local message-passing algorithms risk accounting for the
same evidence multiple times since the evidence from a single source may arrive at a specific node through multiple paths. Figure 5 illustrates several possible network structures. Figure 5 shows a singly connected Bayesian Network (upper left); an example of a Bayesian Network with undirected cycles (loops)(upper right); an undirected graph (a Markov Random Field-MRF) without loops (lower left) and a MRF with multiple loops.

![BN Polytree](image1)  ![Loopy BN](image2)  
![Non-Loopy MRF](image3)  ![Loopy MRF](image4)

**Figure 5 Example Network Topologies**

In complex real world applications, loopy graphical models are common. For example, the Turbo-Codes error-correcting coding/decoding algorithm (mentioned in Chapter 1) is an example of BP applied to a loop graphical model. Bayesian network models for the QMR-DT, a decision-theoretic reformulation of the Quick Medical Reference (QMR) knowledge base (Shwe, Middleton et al. 1991), are extensively loopy and are difficult to solve.
3.2.2 Background on Belief Propagation

Loopy belief propagation is the terminology often used for application of Pearl’s belief propagation algorithm (the polytree algorithm) to networks with loops (Pearl 1988). The BP algorithm is known to converge to the correct (exact) solution for singly connected networks but it is not guaranteed to do so for networks with loops. Pearl had expressed the concern that for loopy networks the beliefs may not converge and even if BP does converge it may not converge to the correct solution. Largely as a result of these concerns, little attention was directed at using the polytree algorithm for loopy networks. This changed with the recognition that the Turbo Codes algorithm (Berrou, Glavieux et al. 1993) for error correcting coding/decoding is an instance of BP in a loopy Bayesian Network (MacKay, McEliece et al. 1998). Turbo Codes have been hailed as the “most exciting and potentially important development in coding theory in many years” (MacKay, McEliece et al. 1998) with near Shannon limit performance. These findings have resulted in a resurgence in interest in applying belief propagation for general probabilistic graphical models. Empirical studies (Frey and MacKay 1998; Murphy, Weiss et al. 1999; Mateescu, Dechter et al. 2002) indicate that BP is a very efficient approximation algorithm and generally converges to the correct solution, and when it does not converge this behavior can be easily detected (Crick and Pfeffer 2003). Actually, BP-like algorithms had even earlier application in error correcting communications. Low density parity check codes were invented along with an iterative probability-based decoding algorithm (the sum-product algorithm) by Gallager (1963). Gallager’s sum-product algorithm is similar to the belief propagation algorithm. However, Gallager’s work was largely ignored until the success of the Turbo Codes algorithms was demonstrated. Subsequently, they were rediscovered by others (Richardson, Shokrollahi et al. 2001; Richardson and Urbanke 2001).

3.3 Detailed Discussion on Probabilistic Graphical Models

This section will discuss the principal types of PGMs including Bayesian Networks and Markov Random Fields and why they have become one of the dominant knowledge
representation and inference techniques in a number of areas including AI, statistical inference and machine learning.

A probabilistic graphical model consists of a graphical model and an associated probability model.

Let $G(\mathcal{V},\mathcal{E})$ be a graph where $\mathcal{V} = (v_1, v_2, \ldots, v_n)$ are the vertices (nodes) of the graph and $\mathcal{E}$ are the edges of the graph. Each node in the graph is associated with a random variable in the probability model. Let $\mathcal{X}$ be the set of random variables $\mathcal{X} = (x_1, x_2, \ldots, x_n)$ that are indexed on the vertices of the graph. The random variables, $x_i$, may be discrete or have continuous distributions. For our purposes we will only consider the discrete case with each $x_i$ limited to a finite number of mutually exclusive (and exhaustive) states $x_i = (x_i^1, x_i^2, x_i^3, \ldots, x_i^{n_i})$ where $n_i$ is the number of states of $x_i$. Furthermore, we will use the symbol $\mathcal{X}$ to refer to both the graphical node and the associated random variable in the remainder of the paper.

A probabilistic graphical model consists of both a structure and parameters (Smyth 1997). The structure of the model is exhibited by (missing) edges in the graphical model and delineates the dependencies (and independencies) that exist among the random variables in the probability model. The edges connecting nodes in the graph represent dependencies between random variables in the probability model and conditional independency relationships are indicated by missing edges. The parameters of the graphical probability model are local conditional probabilities or potential functions and are discussed below.

An edge in a graph will be denoted by $(i, j)$, where $i$ and $j$ are the indices for the two nodes that are connected by the edge. Two nodes are said to be adjacent if there exists and edge that connects them. In a fully connected graph, that is where every node has an edge connecting it with every other node in the graph, all random variables are directly
dependent on each other. At the other extreme, where there are no edges connecting any nodes, all the random variables are independent. Generally, for cases of interest, the structure of the graph falls in between these two extremes.

Two general classes of graphs exist, those where the edges are directed such as for Bayesian Networks and those where the edges are undirected such as for a Markov Random Field. A directed graph is generally indicated for modeling problems where there are identifiable causal relationships among the nodes (random variables). Conversely, where causal relationships are not present (or not easily identifiable) but the associations between random variables appear to be correlated, then an undirected graphical model is suggested.

Undirected graphical models, such as Markov Random Fields (MRFs), have a simple definition of independence: two sets of nodes A and B are conditionally independent given a third set, C, if all paths between the nodes in A and B are separated by a node in C. However, for directed graphical models (without directed cycles), such as Bayesian Networks, the notion of independence is more complicated, and must consider the directionality of the edges.

The probabilistic graphical model is a set of probability distributions defined by a directed or undirected graph. The joint probability distribution is defined by products of functions on subsets of nodes on the graph (Jordan 2004). In general the joint probability distribution is given (using the chain rule) by:

\[
P(\mathcal{X}) = P(x_1, x_2, \ldots, x_n) = P(x_1 | (x_2, \ldots, x_n))P(x_2 | (x_3, \ldots, x_n)) \cdots P(x_{n-1} | x_n)P(x_n) \quad (3.1)
\]

The local dependency relationships for each random variable associated with a node are given by conditional probability tables for directed graphs and by potential functions\(^1\) for undirected graphs. The key feature of probabilistic graphical models is that they allow exploitation of conditional independence relationships in the graph to simplify the

---

\(^1\) Potential functions are non-negative functions related to probabilities.
factorization of the joint probability distribution. The conditional independence relationships allow a much more compact representation of the joint probability distribution. These factors are the local conditional probabilities for directed graphs and (clique) potential functions for undirected graphs.

Let \( \mathcal{N}(x_i) \triangleq \{ x_j \in \mathcal{X} \mid (i, j) \in \mathcal{E} \} \) indicate the neighborhood of \( x_i \). The neighborhood of a node \( x_i \) represents the set of nodes in the graph that are directly connected to \( x_i \) by an edge. The neighborhood of node \( x_i \) consists of the set of nodes that are adjacent to node \( x_i \). Conditioned on the nodes in the neighborhood of \( x_i \) the probability (distribution) of node \( x_i \) is independent of the nodes in the graph outside of its immediate neighborhood.

\[
P(x_i \mid \mathcal{X} \setminus x_i) = P(x_i \mid \mathcal{N}(x_i))
\]

The symbol “\( \setminus \)" indicates that the set of nodes is not to include \( x_i \). Equation (3.2) indicates that the probability of node \( x_i \) is independent of all nodes except those nodes in the neighborhood of \( x_i \), given the values of the nodes in the neighborhood of \( x_i \). \( x_i \) is said to be conditionally independent of these nodes. The notation for this conditional independence relationship is generally shown as \( x_i \perp \mathcal{X} \setminus \mathcal{N}(x_i) \).

As a result of the conditional independence relationships associated with the structure of the graph the joint probability distribution can be factored as:

\[
P(\mathcal{X}) = \prod_{i=1}^{n} P(x_i \mid \mathcal{N}(x_i))
\]

For directed acyclic graphs (DAGs), such as Bayesian Networks, each neighboring node of a node \( x_i \) falls into one of two categories depending on the direction of the edge connecting \( x_i \) with its neighbor. If the edge is directed from a neighbor towards \( x_i \) then the neighboring node is said to be a parent of node \( x_i \). If the edge points from \( x_i \) towards the neighboring node, then the neighboring node is called a child of node \( x_i \). The set of parent
nodes is delineated by \( \text{pa}(x_i) \) or in shorthand form as \( \text{pa}_i \). For DAGs the factorization given in equation (3.2) becomes:

\[
P(x_i | \mathcal{X} \setminus x_i) = P(x_i | \text{pa}_i)
\]

(3.4)

where \( P(x_i | \text{pa}_i) \) is the probability of \( x_i \) conditioned on the parents \( \text{pa}_i \) of \( x_i \).

The factorization of the joint probability distribution that arises from the conditional independence relationships due to the structure of the directed graph results in:

\[
P(\mathcal{X}) = \prod_{i=1}^{n} P(x_i | \text{pa}_i)
\]

(3.5)

Let \( \mathcal{C} \) be a set of cliques of an undirected graph. (Note a clique is a subgraph of the graph \( G \) in which all the nodes are interconnected, that is each node has edges connecting it with all other nodes in the clique. Furthermore a clique \( C \) is maximal if there are no other cliques which are proper subgraph of \( C \)). Let \( \psi_C(x_c) \) be a non-negative potential function associated with each clique \( C \in \mathcal{C} \). The Hammersly-Clifford theorem indicates that the joint probability function \( P(\mathcal{X}) \) satisfies the conditional independence relationships for the graph \( G \), if and only if, \( P(\mathcal{X}) \) is given by the product of the maximal clique potentials (Jaakkola 2001; Jordan and Weiss 2002; Jordan 2004):

\[
P(\mathcal{X}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_C(x_c)
\]

(3.6)

where \( Z \) is a normalization constant function (which assures that \( \sum_X P(\mathcal{X}) = 1 \)) given by:
\[ Z = \sum_{X \in C} \prod_{c \in C} \psi_c(x_c) \]  

(3.7)

and the \( x_c = \{x_i\}_{i \in c} \) represent the set of random variables associated with the nodes that are contained in clique \( c \).

We will generally work with network configurations as shown in Figure 6 where some or all hidden inference nodes are connected to an evidence node. The evidence node may, for example represent a noisy sensor reading (state). With this network configuration it is convenient to rewrite equation (3.6) in a modified form. It is typical to separate the potential functions for the pairwise cliques associated with the \((x_i,y_j)\) node pairs, which represent the “local evidence”, from the cliques formed by the hidden nodes. Let \( \psi_{\mu}(x_i,y_j) \) represent the potential function for the \((x_i,y_j)\) node pair cliques and \( \psi_{\varepsilon}(x_c) \) represent the potentials for the hidden node cliques. This change results in

\[
P(\mathcal{X}) = \frac{1}{Z} \prod_{c \in C} \psi_{\varepsilon}(x_c) \prod_{i} \psi_{\mu}(x_i)
\]  

(3.8)

where we have used the shorthand notation \( \psi_{\mu}(x_i) \) for \( \psi_{\mu}(x_i,y_j) \).

![Figure 6 Example of a Typical Network Configuration](image-url)
3.4 Inference Algorithms for Probabilistic Graphical Models

Computational inference algorithms are most often developed for undirected graphical structures (Smyth 1997). Hence the solutions techniques for directed graphs such as Bayesian Networks, often involve a transformation of the directed graphs to undirected graphs. This is done by first connecting all the parents of each node in the directed graph by undirected edges. This operation is termed marrying the parents. This constructs what is known as a graphical cover for the original directed graph. If the edge directions are also removed then an undirected graph (undirected graphical cover) is obtained which is known as a moral graph (since the parents of a child node are now married). Construction of the moral graph results in the parent-child nodes forming a clique in the undirected graph. Equation (3.5) can be shown to be a special case of Equation (3.6) allowing the application of inference algorithms developed for undirected graphs to be also applied to directed graphs (Jordan and Weiss 2002).

Conversion of a BN or general MRF to a pairwise Markov Random Field considerably simplifies the description and analyses of message-passing (e.g. belief propagation) inference algorithms. A pairwise MRF is an undirected graphical model where the largest clique size is two nodes and hence the potential functions involve at most two adjacent nodes. Any graphical model can be converted into a pairwise MRF by the suitable addition of auxiliary nodes (RVs). Weiss (2000) discusses a technique to convert a general BN into a pairwise MRF. His approach maintains loops (if BN has loops then so does the pairwise MRF). The update rules in the pairwise MRF reduce to Belief Propagation in the original BN and every message in the pairwise MRF can be identified with a Belief Propagation message.

Given a complete joint probability distribution (JPD) over all the random variables in the probabilistic graphical model, all possible inference queries can be obtained by marginalization (summing out over non-query random variables). However the number of unique instantiations of the JPD, scales as $n_s^N$; where $N$ is the number of nodes, and $n_s$ is the number of discrete states for each node. Hence, both the determination of the JPD...
(by direct enumeration of all state combinations) and marginalization have a complexity that is exponential in the number of nodes in the model.

The problem of inference in probabilistic models generally is associated with assessing the posterior probabilities of a set of query variables \( Q \) given a set of evidence variables \( E \) in the graph which have been observed (instantiated).

Two main types of inference tasks are performed using Bayesian Networks 1) belief updating (inference) and 2) belief revision (maximum \( a \) posteriori probability-MAP updating). Belief updating involves the determination of \( P(Q|E) \) – the probabilities of a set of query variables (a subset of all the variables in the network) given the evidence \( E \) (a set of observed (instantiated) values for the network). For a single query variable, \( P(x_i=x_i|E) \) is the posterior marginal probability given the evidence. Belief revision consists of determining the most probable instantiations of a set of variables given the evidence.

The solution methods available for performing the inference calculations in graphical models can be classified at a high level into two categories 1) exact methods and 2) approximate methods. These in turn can be subdivided into additional categories based on the characteristics of the solution technique. Approximate methods can be broadly separated into stochastic sampling algorithms and variational algorithms. A brief summary of the several of the more widely used exact and stochastic inference methods is presented below. However, I will only consider the latter class (variational methods), in-depth since calculating exact solution for the posterior marginals on nodes in a general Bayesian belief network has been demonstrated to be NP-hard (Cooper 1990). The complexity of exact calculation techniques such as variable elimination is exponential in the treewidth. The treewidth is the minimum of the largest of the cliques (minus 1) evaluated over all possible elimination ordering of the nodes in the graph. Even the determination of the elimination ordering which achieves the smallest maximal clique (i.e. the treewidth) has been shown to be in general, NP-hard (Arnborg, Corneil et al. 1987). The calculation of approximate solutions using stochastic simulation techniques
(such as Gibbs sampling/Markov chain Monte Carlo) has also been shown to be, in general, NP-hard (Dagum and Luby 1993).

Guo and Hsu (2002) list the following classes of exact solution techniques for Bayesian Networks:

- **Polytree algorithm (Pearl 1988)** - The Polytree algorithm is a local message-passing belief propagation algorithm that is exact only for singly-connected networks. The polytree algorithm is a generalization of the forwards-backwards algorithm for Hidden Markov Models. When implemented in an iterative manner and applied to graphical models with loops it is referred to as the loopy belief propagation algorithm and is an approximate method.

- **Clustering** – the most popular exact BN solution techniques is the clique tree propagation algorithm (Lauritzen and Spiegelhalter 1988). To avoid double counting of evidence in loopy networks, a common approach is to convert the graph structure into a tree, by clustering nodes together, to form what is called a clique tree (also called a junction tree), and then running a local message-passing algorithm such as some variant of Pearl’s polytree algorithm on this tree. The time complexity of the clique tree propagation algorithm is exponential in the size of the largest clique in the clique tree. This size is called the induced width of the graph. Minimizing the largest clique size is NP-hard.

- **Cut Set Conditioning (Pearl 1988)** – This method works by transforming the network into multiple simpler polytree networks and then using the polytree propagation algorithm. The network is rendered singly connected by instantiating a selected subset of nodes referred to as a loop cut set. Each simple network has one or more variables instantiated to a specific value. The results of each instantiation are computed as a weighted average over the values computed by each polytree. This algorithm has exponential time complexity in the size of the loop cut set. Hence it is important to minimize the loop cut set size. Unfortunately, the loop cut set minimization problem is NP-hard.
• Elimination – Variable elimination/bucket elimination is another popular exact inference technique (Dechter 1999; Cozman 2000). The variable elimination algorithm operates by eliminating variables one by one by summing them out in an efficient order such as to reduce the number of numerical operations (multiplications and summations) required. The optimal elimination ordering minimizes the complexity. However, in general, determining the optimal elimination ordering is NP-hard.

Other less widely used exact inference algorithms that have been developed include arc reversal/node reduction (Shachter 1985; Shachter 1989), symbolic probabilistic inference (Shachter, D'Ambrosio et al. 1990; Li and D'Ambrosio 1994), and differential methods (Darwiche 2000).

Guo and Hsu indicate that for all exact inference methods the time complexity is exponential in the induced width of the graph of the network (the size of the largest clique in the triangulated moral graph). Furthermore, for networks with many loops the induced width is large and exact solutions become intractable.

Guo and Hsu classify the approximate inference techniques into the following categories:

• Stochastic – These methods are the most well know approximation techniques and include probabilistic logic sampling (Henrion 1986), likelihood weighting/evidence weighting (Fung and Chang 1989; Shachter and Peot 1989), importance sampling (Shachter and Peot 1989) and Markov Chain Monte Carlo (MCMC) sampling (e.g. Gibbs sampling, Metropolis sampling, Latin-hypercube stratified sampling) (Geman and Geman 1984; MacKay 1998). The accuracy of these techniques depends on the sample size and is independent of the network structure.
• Model Simplification – these methods rely on procedures to reduce the network complexity until the resulting network structure is sufficiently simplified that exact inference methods can be applied.

• Search-based methods – these techniques assume the majority of the probability mass is contained in small regions of the probability space and the search algorithms look for the high probability instantiations which are used to approximate the probability distribution.

• Loopy belief propagation – an application of Pearl’s polytree algorithm in an iterative manner to networks with loops. As mentioned previously, this risks the double counting of local evidence. Weiss (2001) has shown that for certain conditions (e.g., a single loop), events are double counted equally, and hence cancel to give the correct (exact) maximum a posteriori (MAP) assignments even though the numerical values of the beliefs are wrong. Loopy belief propagation is equivalent to use of the Bethe approximation in the variational methods category of algorithms.

• Variational methods - Variational methods involve recasting the inference problem as a statistical physics/thermodynamics problem defined on a graph and minimization of certain approximations (mean field, Bethe, Kikuchi) to the Gibb’s free energy. Yedidia, Freeman et al (2000) have shown that the zero gradient points of the Bethe free energy approximation correspond to the fixed points of Pearl’s BP algorithm.

3.4.1 Message-Passing Belief Propagation Algorithms

This section provides a brief summary of message-passing algorithms beginning with Pearl’s (1988) belief propagation (BP) algorithm which is exact for singly-connected Bayesian Networks. BP is similar to the forward-backward algorithm used for the solution to Hidden Markov Models. BP is a decentralized iterative algorithm that operates by transmitting messages between nearby nodes (variables) in the probabilistic
graphical model. Each node in the graphical model acts as a processor which can communicate only with its nearest neighbors. For a Bayesian Network the nearest neighbors correspond to the node’s parents and children. The processor associated with each node maintains knowledge of the conditional probabilities of its states conditioned on its parents states \( P(x_i|p_{ai}) \). If a node has no parents then it maintains knowledge of its marginal probability \( P(x_i) \). When a node processor is activated, it reads all incoming messages from its parents (\( u_k \)’s) and children (\( y_j \)’s), updates its belief, and sends out new messages to its parents and children. Messages received from the parents of node \( x_i \) are designated by \( \pi_{u_i,x}(u_k) \) and from its children by \( \lambda_{y_j,x}(x_i) \). The \( \pi_{u_i,x}(u_k) \) messages are in the form of lists of probabilities (hence \( \pi \) messages), one for each state value in \( u_k \). These represent the belief in the probability of each state in \( u_k \) that node \( u_k \) maintains based on all the evidence it has received. The \( \lambda_{y_j,x}(x_i) \) messages consists of a list of nonnegative numbers (likelihoods hence \( \lambda \) ) for each state value in \( x_i \). These represent the beliefs in the probabilities of the evidence that \( y_j \) has received conditioned on each state of \( x_i \).

Each node \( x_i \) computes a belief, \( b_{x}(x_i) = P(x_i | E) \), regarding the state of the RV associated with the node by combining messages it has received from its neighboring nodes. \( E \) represents the evidence; that is the set of observed (instantiated) nodes in the network.

\[
b_{x}(x_i) = a\lambda_{x}(x_i)\pi_{x}(x_i) \tag{3.9}
\]

where:

\[
\lambda'_{x}(x_i) = \lambda_{x}(x_i)\prod_{j}^{\lambda_{y_j}(x_i)} \tag{3.10}
\]

and:

\[
\pi'(x_i) = \sum_{u_k} P(x_i = \hat{x}_i | u_k = \hat{u}_k)\prod_{m}^{\pi'_{x}(u_m)} \tag{3.11}
\]

The messages that \( x_i \) passes to its set of parents \( u_k \) :
\[
\lambda_{X}^{t+1}(u_k) = \alpha \sum_{x_i} \lambda_{x}^{t}(x_i) \sum_{u_{m \neq k}} P(x | u) \prod_{m \neq k} \pi_{X}^{t}(u_m)
\]  
(3.12)

and the message \(x_i\) transmits to its child \(y_j\) is:

\[
\pi_{y_j}^{t+1}(x_j) = \alpha \pi_{y}^{t}(x_j) \lambda_{y}^{t}(x_j) \prod_{n \neq j} \lambda_{y_j}^{t}(x_j)
\]  
(3.13)

The messages contain all the information (evidence) necessary to locally evaluate the posterior marginals for each variable.

**Message-passing Algorithm for Pairwise MRF**

Let \(x_i\) and \(x_j\) be neighboring nodes in the pairwise MRF. \(m_{ij}(x_j)\) is the message that node \(x_i\) sends to node \(x_j\). \(\phi_{i}(x_i)\) is the potential at node \(x_i\) (representing the local evidence available at node \(x_i\)). \(\phi_{ij}(x_i,x_j)\) is the pairwise potential relating node \(x_i\) and \(x_j\). \(b_{i}(x_i)\) is the marginal belief at node \(x_i\) and \(b_{ij}(x_i,x_j)\) is the pairwise joint marginal belief for nodes \(x_i\) and \(x_j\).

The BP sum-product update rules are:

\[
m_{y}^{t+1}(x_j) = \alpha \sum_{x_i} \phi_{y}^{t}(x_i,x_j) \phi_{j}^{t}(x_j) \prod_{k \neq j} m_{y}^{t}(x_j)
\]  
(3.14)

\[
b_{i}^{t}(x_i) = \alpha \phi_{i}^{t}(x_i) \prod_{k} m_{i}^{t}(x_i)
\]  
(3.15)

\[
b_{ij}^{t}(x_i,x_j) = \alpha \phi_{y}^{t}(x_i,x_j) \prod_{k \neq j} m_{i}^{t}(x_i) \prod_{l \neq i} m_{y}^{t}(x_j)
\]  
(3.16)
3.5 Variational Approximation Methods

Variational methods involve recasting the probabilistic inference problem as a statistical physics/thermodynamics problem defined on a graph and minimization of certain approximations (mean field, Bethe, Kikuchi) to the Gibb’s free energy (variational free energy) subject to certain constraints.

The Gibbs free energy is a function of the full joint probability distribution $G = G(P(\mathcal{X}))$ and is given by

$$G = U - TS$$

(3.17)

where:

- $G$ = Gibb’s free energy of a system
- $U$ = average internal energy
- $T$ = temperature
- $S$ = entropy

The entropy is given by:

$$S = -\sum_{\mathcal{X}} P(\mathcal{X})\log(P(\mathcal{X}))$$

(3.18)

and the internal energy (the average energy, averaged over all possible states of the system)

$$U = \sum_{\mathcal{X}} P(\mathcal{X})E(\mathcal{X})$$

(3.19)

Substituting equations (3.18) and (3.19) into equation (3.17) yields:

$$G = \sum_{\mathcal{X}} P(\mathcal{X})E(\mathcal{X}) + T\sum_{\mathcal{X}} P(\mathcal{X})\log(P(\mathcal{X}))$$

(3.20)
Minimization of the exact Gibbs free energy with respect to $P(\mathcal{X})$ (and the incorporation
of a LaGrange multiplier to enforce the constraint that $\sum_{\mathcal{X}} P(\mathcal{X}) = 1$) results in
Boltzmann’s law for computing the joint probability distribution:

$$P(\mathcal{X}) = \frac{1}{Z} \exp\left\{ -E(\mathcal{X}) / T \right\}$$

(3.21)

where

$$Z = \sum_{\mathcal{X}} \exp\left\{ -E(\mathcal{X}) / T \right\}$$

(3.22)

is a normalization constant (the partition function in statistical physics).

Substituting equation (3.21) into the Gibbs free energy equation (3.20) yields:

$$G = F_{\text{Helmholtz}} = -T \log Z$$

(3.23)

where $F_{\text{Helmholtz}}$ is the Helmholtz free energy giving the value of the Gibbs free energy
when the joint probability distribution is at its correct value (the system is in
equilibrium).

Yedidia et al. (2002) note that when one is working with a joint probability distribution
for a non-physical system Boltzmann’s law can be viewed as defining the energy $E(\mathcal{X})$
for the system where the temperature $T$ can be chosen arbitrarily since it is simply a scale
factor on the units of energy. Consequently, it is generally convenient to set the
temperature $T = 1$.

Minimizing the exact variational free energy with respect to $P(\mathcal{X})$ is generally intractable.
Hence, approximations to the variational (Gibb’s) free energy have been sought.
Variational methods posit a family of approximating probability distributions and allow the optimization algorithm to select the best member of the family of distributions as a solution to the problem (Jordan and Weiss 2002). We will denote the family of approximating distributions as \( \{b(x)\} \). Substituting \( b(x) \) into equation (3.20) for \( P(x) \) and setting \( T = 1 \) yields:

\[
G_v = \sum_x b(x)E(x) + \sum_x b(x)\log(b(x)) \quad (3.24)
\]

Solving the Boltzmann equation (3.21) for \( E(x) \) yields a definition for the energy of the system:

\[
E(x) \triangleq -\log b(x) - \log Z \quad (3.25)
\]

where again \( T = 1 \). Substituting equation (3.25) into (3.24) yields the final form for the variational (Gibbs) free energy:

\[
G_v = \sum_x b(x)(-\log P(x) - \log Z) + \sum_x b(x)\log(b(x)) \quad (3.26)
\]

\[
G_v = -\sum_x b(x)\log P(x) + \sum_x b(x)\log b(x) - \sum_x b(x)\log Z \quad (3.27)
\]

Since \( Z \) is a constant and \( b(x) \) is a probability distribution the last term simplifies to just \( \log Z \) and equation (3.27) becomes:

\[
G_v = -\sum_x b(x)\log P(x) + \sum_x b(x)\log b(x) - \log Z \quad (3.28)
\]

Note that when the approximating distribution \( b(x) \) is equal to the exact distribution \( P(x) \) then the first two terms on the right hand side of equation (3.28) cancel and the
The variational free energy is equal to the Helmholtz free energy (see equation (3.23)). The first two terms can be grouped together and equation (3.28) rewritten as:

\[
G_v = \sum_{\mathcal{X}} b(\mathcal{X}) \log \frac{b(\mathcal{X})}{P(\mathcal{X})} - \log Z = \sum_{\mathcal{X}} b(\mathcal{X}) \log \frac{b(\mathcal{X})}{P(\mathcal{X})} + F_{\text{Helmoltz}}
\] (3.29)

The first term on the right hand side is in the form of a Kullback-Leibler divergence which is written as \([KL(b||p)]\) and is a distance measure for how near \(b(\mathcal{X})\) is to \(P(\mathcal{X})\). It is known that the Kullback-Leibler divergence is always non-negative and is zero if, and only if, \(b(\mathcal{X}) = P(\mathcal{X})\). Hence, \(G_v \geq F_{\text{Helmoltz}}\), with the equality holding only for \(b(\mathcal{X}) = P(\mathcal{X})\). Minimization of \(G_v\) can be as difficult as doing exact inference unless one chooses approximate forms for \((b(\mathcal{X}))\) that are readily minimized.

### 3.5.1 Mean Field Approximation

The simplest variational approximation is the mean-field approximation where the family of distributions allowed for \(b(\mathcal{X})\) is restricted to:

\[
b(\mathcal{X}) = b_{MF}(\mathcal{X}) = \prod_{i=1}^{n} b_i(x_i)
\] (3.30)

Hence, the joint distribution is factored into the product of the individual distributions on each random variable.

Substituting \(b(\mathcal{X})\) in equation (3.30) for \(P(\mathcal{X})\) in equation (3.28) yields:

\[
G_{MF}(b_i) = -\sum_{\mathcal{X}'} [\prod_i b_i(x_i)] \log P(\mathcal{X}') + \sum_{\mathcal{X}'} [\prod_i b_i(x_i)] \log \prod_i b_i(x_i) - \log Z
\] (3.31)
Inserting the factored form of $P(\lambda)$ from equation (3.8) into equation (3.31) and rearranging, grouping and performing some algebraic manipulations yields:

$$G_{MF}(b_i) = -\sum_C \sum_{x_C} \log \psi_c(x_c) \prod_{i \in C} b_i(x_i) + \sum_j \sum_{x_j} b_j(x_j) \{ \log b_i(x_i) - \log \psi_i(x_i) \}$$

(3.32)

with the constraint $\sum_{x_i} b_i(x_i) = 1$.

Setting the derivative of equation (3.32) wrt $b_i(x_i)$ equal to zero yields:

$$b_i(x_i) = \beta \exp \left( \sum_C \sum_{x_C \neq i} \log \psi_c(x_c) \prod_{j \in C, j \neq i} b_j(x_j) \right)$$

(3.33)

where $\beta$ is a normalization constant such that the constraint $\sum_{x_i} b_i(x_i) = 1$ is satisfied.

For a pairwise MRF equation (3.32) becomes

$$G_{MF}(b_i) = -\sum_g \sum_{x_i,x_j} \log \psi_{ij}(x_i,x_j) b_i(x_i)b_j(x_j) + \sum_J \sum_{x_J} b_J(x_J) \{ \log b_i(x_i) - \log \psi_i(x_i) \}$$

(3.34)

Equation (3.33) represents an approximate message-passing inference algorithm. The approximate marginals (values for $b_i(x_i)$) for all nodes are initialized and then updated one node at a time in an iterative manner. Since the right hand side of equation (3.33) involves only cliques that $x_i$ is in (i.e. nodes in the neighborhood of $x_i$), $b_i(x_i)$ is updated locally based on information from its immediate neighbors. The mean field approximation depends upon approximate marginals at single nodes. The next level of approximation involves approximation of the marginals at individual nodes and on cliques. This approximation is called the Bethe free energy approximation.
3.5.2 Bethe Approximation

The Bethe free energy approximation (Bethe 1935) improves upon the mean-field approximation by taking into account correlations between nearest neighbor “sites”. The Bethe approximation is the simplest of a hierarchy of approximations that account for correlations among progressing larger clusters of sites collectively termed cluster variational methods (Pretti and Pelizzola 2003). The variational parameters in the Bethe approximation are the single site and nearest neighbor pair probability distributions. Recently, it has been shown that there is a relationship between the belief propagation algorithm developed by Pearl (1988) for solving inference problems in general probabilistic graphical models and the Bethe approximations to the variational free energy (Gibbs’ free energy) (Yedidia, Freeman et al. 2000; Aji and McEliece 2001). It is known that the belief propagation algorithm is exact for singly-connected graphs (that is, a graph without loops) (Pearl 1988). Not surprisingly, the Bethe approximation is also exact for singly-connected graphical structures (Pretti and Pelizzola 2003). Yedidia et al. show that BP can only converge to a stationary point (zero gradient point) of the Bethe free energy and they have found that belief propagation corresponds to iterative descent down an associated Bethe free energy. Recent work has shown that belief propagation/Bethe approximation work surprisingly well on many graphical structures with loops (Frey and MacKay 1998; MacKay, McEliece et al. 1998; Murphy, Weiss et al. 1999; Mateescu, Dechter et al. 2002).

For the Bethe approximation the family of distributions allowed for \( b(\mathcal{X}) \) is given by the product of individual node marginal beliefs and the joint pairwise marginal beliefs of adjacent nodes:

\[
b(\mathcal{X}) = b_{\text{Bethe}}(\mathcal{X}) \approx \prod_{\langle ij \rangle} b_{ij}(x_i, x_j) \prod_{i=1}^{n} b_i(x_i)^{1-n_i} \tag{3.35}
\]

The Bethe approximation to the variational free energy is given by (Jordan and Weiss 2002):
where \( n_i \) = the number of cliques that node \( i \) belongs to.

Lagrange multipliers are utilized to enforce the constraint that the clique joint marginal beliefs marginalize down to the individual node beliefs \( \sum_{x_c} b_c(x_c) = b_i(x_i) \).

For a pairwise MRF equation (3.36) becomes:

\[
G_{\text{Bethe}}(b_i, b_{ij}) = \sum_{ij} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \frac{b_{ij}(x_i, x_j)}{\psi_{ij}(x_i, x_j)\psi_i(x_i)\psi_j(x_j)} + \sum_i (n_i - 1) \sum_{x_i} b_i(x_i) \log \frac{b_i(x_i)}{\psi_i(x_i)} \tag{3.37}
\]

3.5.3 General Kikuchi Approximations

The Bethe approximation is the simplest form of the Kikuchi “cluster variational method” (Kikuchi 1951). More generally, in Kikuchi approximations, the free energy is approximated as a sum of the free energies of basic clusters of nodes (which may be overlapping), minus the free energy of over-counted cluster intersections, minus the free energy of the over-counted intersections of intersections, and so on. For the Bethe approximation, the basic clusters involve only linked pairs of nodes (Yedidia, Freeman et al. 2001). Higher level Kikuchi approximation involve interactions among more extensive clusters of nodes in addition to pair-wise basic clusters as in the Bethe approximation. Generally, the more nodes included in the basic clusters the better the approximation. However, as the basic cluster sizes are increased, the computational complexity of the algorithm also increases, so there exists a tradeoff between the level of approximation and computational tractability. Yedidia, Freeman et al. (2001) have introduced a general belief propagation (GBP) algorithm that extends Pearl’s belief propagation algorithm by allowing minimization of arbitrary Kikuchi free energies (different sizes for the basic clusters). This algorithm has the property that it reduces to
Pearls algorithm when the assumptions underlying the Bethe approximation are implemented (i.e. pair-wise basic clusters).

The Kikuchi approximation to the variational free energy given by (Yedidia, Freeman et al. 2001):

\[
G_{\text{Kikuchi}} = \sum_{r \in R} c_r \left( \sum_{x_r} b_r(x_r) E_r(x_r) + \sum_{x_r} b_r(x_r) \log b_r(x_r) \right)
\]  

(3.38)

where \( r \) is a region in the set of regions \( R \), \( x_r \) is the state of the nodes in region \( r \), \( b(x_r) \) is the belief in \( x_r \), \( c_r \) is the over-counting number of region \( r \) given by \( c_r = 1 - \sum_{s \in \text{super}(r)} c_s \), and \( \text{super}(r) \) is the set of all super-regions of \( r \).

Figure 7 presents examples of the basic node clusters used in the Mean Field, Bethe and Kikuchi (4 node cluster) approximations for a square lattice pairwise Markov Random Field.

3.6 Double Loop Iterative Algorithms

A number of researchers have recently developed double loop iterative algorithms that are guaranteed to converge to the minima of the approximate Gibb’s free energy (Bethe
Yuille (2001; 2002; Yuille and Rangarajan 2003) introduced a class of provably convergent iterative algorithms that exploit the structure of the approximate free energy function (Bethe or Kikuchi) and are alternatives to BP and generalized belief propagation. Their algorithm, called the Concave Convex Procedure (CCCP), operates by decomposing the free energy function into concave and convex parts, and introduces a double loop iteration scheme where the inner loop determines the constraints on the beliefs and the outer loop iterates the update rules (subject to the constraints) for the beliefs such that the free energy is reduced during each iteration. Heskes, Albers et al. (2003) also developed a class of double loop algorithms, in which the inner loop corresponds to constrained minimization of a convex bound on the Kikuchi free energy and the outer iteration loop recalculates the bound. Teh and Welling (2001) introduced the Unified Propagation and Scaling Algorithm (UPS) which shares certain similarities with the Heskes, Albers et al. algorithm. However, instead of bounding the portions of the concave entropy, UPS clamps some of them (clamps a number of hidden nodes to their current marginals) during the iteration. The following discussion presents the derivation of the CCCP algorithm from Yuille (2002).

The Bethe free energy for a pairwise Markov Random Field is:

$$G_{\text{Bethe}}(\{b_{ij}, b_i\}) = \sum_{i,j \neq j} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \frac{b_{ij}(x_i, x_j)}{\phi_{ij}(x_i, x_j)} - \sum_i (d_i - 1) \sum_{x_i} n_i(x_i) \log \frac{b_i(x_i)}{\phi_i(x_i)}$$  \hspace{1cm} (3.39)

where $\phi_{ij}(x_i, x_j) = \psi_{ij}(x_i, x_j) \psi_i(x_i) \psi_j(x_j)$

Separating equation (3.39) into concave and convex parts yields:

$$E_{\text{vec}} = \sum_{i,j \neq j} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \frac{b_{ij}(x_i, x_j)}{\phi_{ij}(x_i, x_j)} + \sum_i \sum_{x_i} b_i(x_i) \log \frac{b_i(x_i)}{\phi_i(x_i)}$$  \hspace{1cm} (3.40)
\[ E_{\text{cave}} = -\sum_i n_i \sum_{x_i} b_i(x_i) \log \frac{b_i(x_i)}{\phi_i(x_i)} \] (3.41)

Now we have an energy function of the form:

\[ E(\vec{z}) = E_{\text{ver}}(\vec{z}) + E_{\text{cave}}(\vec{z}) \] (3.42)

where

\[ \vec{z} = (b_q(x_i, x_j), b_i(x_j)) \] (3.43)

Taking the derivative of (3.42) and setting the result to zero yields:

\[ \nabla E(\vec{z}) = \nabla E_{\text{ver}}(\vec{z}) + \nabla E_{\text{cave}}(\vec{z}) = 0 \] (3.44)

Hence;

\[ \nabla E_{\text{ver}}(\vec{z}) = -\nabla E_{\text{cave}}(\vec{z}) \] (3.45)

Yuille has proven that an iterative algorithm of the form

\[ \nabla E_{\text{ver}}(\vec{z}^{t+1}) = -\nabla E_{\text{cave}}(\vec{z}^t) \] (3.46)

will monotonically decrease the energy as a function of iterative time (t) and converge to a minimum of \( E(z) \).

However, the variables represented by \( \vec{z} = (b_q(x_i, x_j), b_i(x_j)) \) are estimates of probability distributions, hence they must satisfy a number of constraints including normalization and consistency (marginalization).

The normalization constraints are given by:

\[ \sum_{x_p, b_q} b_{pq}(x_p, x_q) = 1 \quad \forall p, q : p > q \quad \text{and} \quad (3.47) \]
\[ \sum_{x_p} b_p(x_p) = 1 \quad \forall p \]  \hspace{1cm} (3.48)

and the marginalization constraints by:

\[ \sum_{x_p} b_{pq}(x_p, x_q) = b_q(x_q) \quad \forall p, q, x_q : p > q \quad \text{and} \]  \hspace{1cm} (3.49)

\[ \sum_{x_q} b_{pq}(x_p, x_q) = b_p(x_p) \quad \forall p, q, x_p : p > q \]  \hspace{1cm} (3.50)

The messages sent from one node to another are of the form

\[ m_j(x_j; t+1) = c_{ij} \sum_{x_i} \phi_j(x_j, x_i) \phi_i(x_i) \prod_{k \in j} m_k(x_i, t) \]  \hspace{1cm} (3.51)

where

\[ m_j(x_j; t+1) \] are the messages sent from node \( i \) to neighboring nodes \( j \) at iterative time \( t+1 \)

\[ m_k(x_i; t) \] are the messages sent from neighboring nodes \( k \) to node \( i \) during prior iterative time \( t \)

\( c_{ij} \) are normalization constants

These messages correspond to Lagrange multipliers which enforce the normalization and marginalization constraints on the beliefs. The update equations for the beliefs (the approximate marginal probabilities for node \( i \) and the joint marginal probabilities at neighboring nodes \( i,j \) ) are given by

\[ b_i(x_i; t) = c_i \phi_i(x_i) \prod_{j \in N(i)} m_j(x_i, t) \]  \hspace{1cm} (3.52)

The constraints are imposed by forming the Lagragian:
\[ L = G_{\text{Bethe}}(\{b_y, b_i\}) + \sum_{i \neq j} \gamma_{ij} \left\{ \sum_{x_i, x_j} b_y(x_i, x_j) - 1 \right\} \]

\[ + \sum_{i, j \neq i} \sum_{x_j} \lambda_{ij}(x_j) \left\{ \sum_{x_i} b_y(x_i, x_j) - b_j(x_j) \right\} \]

\[ + \sum_{i, j \neq i} \sum_{x_i} \lambda_{ji}(x_i) \left\{ \sum_{x_j} b_j(x_i, x_j) - b_i(x_i) \right\} \]

\[ + \sum_{i} \gamma_i \sum_{x_i} \{b_i(x_i) - 1\} \]  
(3.53)

and differentiating with respect to \( b_y \) and \( b_i \). Note that the \( \gamma_{ij} \) terms impose the normalization constraints on the pairwise marginal probabilities and the \( \lambda_{ij} / \lambda_{ji} \) terms enforce the marginalization constraints.

Taking the partial derivative of \( L \) with respect to \( b_y \) and equating to zero yields:

\[ \ln b_y(x_i, x_j) = \ln \phi_y(x_i, x_j) - \lambda_y(x_j) - \lambda_{ji}(x_i) - \gamma_{ij} - 1 \]  
(3.54)

or

\[ b_y^{*+l}(x_i, x_j) = \phi_y(x_i, x_j) e^{-\lambda_y(x_j)} e^{-\lambda_{ji}(x_i)} e^{-\gamma_{ij}} e^{-1} \]  
(3.55)

Taking the partial of \( L \) with respect to \( b_i \) and setting to zero yield:

\[ (n_i - 1)(\ln b_i(x_i) + 1) = (n_i - 1) \ln \phi_i(x_i) - \sum_{j \in N(i)} \lambda_{ji}(x_i) + \gamma_i \]  
(3.56)

or

\[ b_i(x_i) = \phi_i(x_i) e^{-1} e^{\gamma_i} \left( \frac{b_i(x_i)}{\phi_i(x_i)} \right) e^{\sum_{j} \lambda_{ji}(x_j)} e^{\gamma_i} \]  
(3.57)

Note: the last term in equation (3.57) is generally subsumed into the normalization constant for \( b_i \).
Equations (3.55) and (3.57) constitute the update equations for the beliefs in the outer loop of the CCCP algorithm subject to the constraint coefficient terms $\lambda_{ij}$, $\lambda_{ji}$ and $\gamma_g$, which are determined in an inner iteration loop using:

$$e^{y_{pq}(r+1)} = \sum_{x_p, x_q} \phi_{pq}(x_p, x_q) e^{-\lambda_{pq}(x_p, x_q)} e^{-\lambda_{qp}(x_q, x_p)}$$  \hspace{1cm} (3.58)

$$e^{2\lambda_{pq}(x_p; t+1)} = \frac{\sum_{x_p} \phi_{pq}(x_p, x_q) e^{-\lambda_{pq}(x_p, x_q)} e^{-\gamma_{pq}(x_q)} \varphi_q(x_q) e^{\gamma_{pq}(x_q)}}{\varphi_q(x_q) e^{\gamma_{pq}(x_q)} \sum_{x_q} e^{\gamma_{pq}(x_q)}}$$  \hspace{1cm} (3.59)

$$e^{2\gamma_{pq}(x_p; t+1)} = \frac{\sum_{x_q} \phi_{pq}(x_p, x_q) e^{-\lambda_{pq}(x_p, x_q)} e^{-\gamma_{pq}(x_q)} \varphi_p(x_p) e^{\gamma_{pq}(x_p)}}{\varphi_p(x_p) e^{\gamma_{pq}(x_p)} \sum_{x_p} e^{\gamma_{pq}(x_p)}}$$  \hspace{1cm} (3.60)

Simulations performed by Yuille indicate that the CCCP algorithms are stable and converge quickly (convergence speeds similar to BP/GBP). Also the solutions obtained with the CCCP algorithms for the Bethe or Kikuchi free energy approximations are as good, or slightly better than, those found by BP or GBP respectively (when BP/GBP converged). Note that the solutions obtained by BP/GBP are generally very accurate and failure to converge is their major error mode.
Chapter 4 Research Approach and Methods

This chapter presents the research approach and techniques/methods that have been considered, and selected, for performing the research and why this approach was appropriate to address the research problem. This chapter will discuss how simulated sensor networks topologies and characteristics are to be determined and what types of graphical models (e.g., Bayesian Belief networks, Markov Random Field) were utilized in modeling these simulated sensor networks. The principal simulation parameters (independent variables such as the number of nodes, target density, etc.) and the principal outcomes of interest (e.g., solution quality, communications and energy costs) are also identified.

4.1 Simulation Environment

A node level simulator typically has the following components (Zhao, Liu et al. 2003):

- Sensor node model – modeling of the communications stack, sensor behaviors (sensor noise), and operating system services
- Communications model – simulates communication at some level of abstraction (e.g. at the hardware level modeling RF propagation and attenuation, delays, data collision and interference or at a high level through incorporation of bit-wise or packet-wise error rates)
- Physical environment model – simulation of the environmental parameters of interest
- Statistical and visualization capability – presentation of simulation results including global properties; nodal connectivity; communication routes, delays and error rates; environment phenomena and dynamics; node sensor readings and state estimations; and network lifetime indicators such as node energy remaining

What is the proper simulation environment in which to test algorithm behavior? This is a critical question. The most realistic environment is obviously to use the actual physical components and construct a real sensor network. The inference and information fusion
algorithms then could be coded (in the network/processor operating system language) and
installed directly on the network hardware. There are a number of disadvantages to this
approach, however. First, procurement of a large number of sensor/processing nodes and
their deployment as a sensor network represents a significant cost in both hardware and
labor. Second, much effort would be required in maintenance and reconfiguring the
network into different topologies and architectural arrangements. Third, since the
network processing and communications will be performed on multiple distributed
physical platforms it will be difficult to intervene to debug, locate bottlenecks and
limitations, and to assess algorithm performance. Fourth, scalability would be difficult to
assess since the number of sensor nodes would be limited to relatively small numbers.

Another approach would be to simulate the network at a high level of abstraction on a
PC/workstation using a common OS (Windows XP/Linux) and high level language such
as C/C++ or in a specialized environment such as Matlab. This approach has the
advantage of allowing the entire network simulation to be performed on a single
platform; hence simplifying debugging and the evaluation of the performance of various
algorithms. However, it may abstract too far away from the actual physical system and
hide critical interactions among the environment, sensor hardware and software,
communications, and inference and fusion algorithms. For example, it may obscure the
effects of limited node processor memory (RAM/ROM), timing and latency issues,
communication issues (transmission/reception errors and interference, asynchronous
communication) and extensive concurrent interactions with the physical environment.

A third approach falls somewhere in between the above two. This approach involves
using simulation software that allows low level simulation of the actual sensor node
hardware and software using the actual software that runs on the physical network. This
approach has a number of desirable features including close modeling of the coupled
interactions of sensors, signals from the environment, network communications and
processing. This approach also allows rather rapid and relatively straightforward changes
to network parameters such as number of nodes, network architectures and
communications topologies.
A number of simulation environments are being developed for use in sensor network research, design and evaluation. These include simulation environments that focus on low level simulation of sensor node behavior such as the TOSSIM simulation environment (Levis, Lee et al. 2003) developed by UC Berkeley for sensor networks utilizing the Berkeley MICA mote sensor nodes and the TinyOS operating system and TOSSF (Perrone and Nicol 2002) developed at Dartmouth University, simulation environments that are directed at simulating wireless communications and protocols includingSensorSim(UCLA) (Park, Savvides et al. 2000) and SensorSimII (Ulmer) and simulation environments more focused on application development and assessment such as SENS (Sundresh, Kim et al. 2004), Siesta (Ledeczi, Maroti et al.) and EmSim/EmStar (Girod, Elson et al. 2004; Girod, Stathopoulos et al. 2004).

TOSSIM was developed to simulate sensor networks running TinyOS on Berkeley motes. TOSSIM was designed to run the actual software code that runs on the Berkeley mote hardware. TOSSIM models the wireless sensor network as a directed graph where each vertex in the graph is a hardware sensor node and each directed edge connecting the vertices represents a communications pathway. Communications errors along the edges are modeled by a stochastic bit-wise error probability. There is a visualization package (TinyViz) to support TOSSIM simulator which provides features for controlling and viewing the simulation. TinyViz facilitates actions such as changing ADC values (e.g. sensor readings), communication error rates and control messages. There are several limitations with TOSSIM. One of the most significant for our purposes is the requirement that all simulated nodes run the same code, hence limiting the simulation to networks of homogenous nodes. Another limitation of both TOSSIM and TOSSF is their tight coupling to TinyOS and the Berkeley mote hardware platforms.

SensorSim, developed at UCLA, provides features for modeling sensor networks including: sensing channel and sensor models, models for batteries, protocol stacks for wireless microsensors, facilities to create environmental scenarios, and the ability to integrate simulated sensor nodes with real physical nodes. A major impediment, however
to using SensorSim is the limited availability of the software components and user support.

SENS (Sensor, Environment and Network Simulator), developed at the University of Illinois, Urbana-Champaign, allows application developers and researchers to execute the same source code on simulated sensor nodes that runs on the real sensor nodes. SENS allows the simulation of the operation and performance of networks comprised of heterogeneous sensors and provides diagnostic facilities such as energy utilization. Several models for communication errors (at the packet or bit level) are available including a traffic-dependent error rate model and a collision-based model. SENS has a relatively detailed model for the environment which is modeled as a two dimensional grid of tiles (representing concrete, grass, and walls) each with different RF attenuation and acoustic properties.

EmSim, also developed at UCLA, provides simulation of the actual coding written in the EmStar framework. EmSim/EmStar, unlike TOSSIM, allows simulation of networks comprised of heterogeneous nodes (for example motes and larger microservers). It also provides a unified set of tools for capturing, viewing, and analyzing debugging information during simulations.

I downloaded and performed a limited assessment and review of the EmSim/EmStar simulation environments for potential use in this research. I selected EmSim/EmStar for review since I was particularly interested in an application-focused simulation environment that allows use of real mote/microprocessor coding, supports simulation of networks of heterogeneous sensors, and allows the assessment of inference/fusion algorithms in energy and resource constrained networks.

My review and assessment suggested that this software is still in a state of development and the documentation is relatively limited and incomplete. I judged that the effort required to learn and potentially modify the software for use in this research would be
substantial and outweighed the potential advantages (discussed above). Consequently, I decided to develop the simulation test environment within the Matlab environment.

4.2 Simulation Problem

The canonical problem to be used for examination of the performance of advanced approximation algorithms in distributed wireless sensor networks is the data association problem for localization and tracking problem. This problem was selected for a number of reasons. Firstly, localization and tracking is a required capability in many sensor network applications. Secondly, the data association problem exhibits many of the central challenges to distributed inference and information fusion including how to dynamically configure collaborative sensor node groups, and how to perform collaborative distributed processing, information sharing and information fusion. Thirdly, the localization and tracking problem spans a broad spectrum of complexity, from locating a single stationary target through locating and tracking multiple targets with multiple sensors.

Figure 8 illustrates the overall localization and tracking cycle. At the beginning of each measurement cycle, estimates of the position for each previously identified target at the next measurement time are generated by a tracking algorithm using current and prior target position estimates (Prediction step). Each sensor then generates measurements for all targets within its coverage range (Observation Step). Each measurement for each sensor is then associated with a specific target during the Association/Correlation step. Finally, all measurements (from different sensors) associated with specific targets are combined to generate the current target position estimates (Estimation step).

The principal area of concentration of this dissertation is in the Association/Correlation phase of the localization and tracking problem.
Following Chen, Wainwright et al. (2003) I constructed an $M \times M$ two dimensional grid onto which $N$ sensors are placed. Each sensor has a specified detection range with all sensors having the same range. The detection range can be varied in the experiments to increase or decrease the extent of coverage overlap among the sensors. A set of targets are then added to the grid. In the simulation matrix, the number of targets (target densities) can be varied. Several options are available to distribute the targets. They can be randomly distributed within the entire grid or subsets of targets can be randomly distributed within the sensing range of specific sensors. The latter case allows simulation of high densities of targets within the sensing range of individual sensors.

It is assumed that the predicted target position follows a normal distribution with a covariance matrix $\Sigma = \sigma^2 I$ and with mean value equal to the mean predicted position.
The value of the predicted target position covariance matrix can be varied to assess the impact of uncertainties is predicted target position on algorithm performance. The true target position is then determined by sampling from the predicted target position distribution. The sensor measurement error will also be characterized by normal distribution with covariance matrix \( \Lambda = \lambda^2 I \). For targets with a true position within the detection range of each sensor a target position measurement will be generated by sampling from the measurement distribution (with mean value equal to the actual target position and covariance \( \Lambda \)). Again, the value of the covariance matrix \( \Lambda \) can be varied to assess algorithm performance.

4.3 Probabilistic Graphical Model

Although in principle, a Bayesian Network could be constructed as the graphical model for this problem, the constraints (for the multitarget/multisensor data association problem) that only one target can be associated with a particular sensor measurement and that a particular measurement can be associated with only a single target would require the introduction of a complicated set of deterministic nodes to assure these constraints are met. A more direct approach is to model this problem using a Markov Random Field (MRF) where each node in the network represents a sensor and the random variable associated with the node models the possible associations of sensor measurements and targets within the range of the sensor. The states of the random variable represent all possible combinations of the associations of specific sensor measurements and targets.

For example, for a sensor \( s_i \) with three targets \([x_{i1}, x_{i2}, x_{i3}]\) within its sensing range and which has generated three measurements \([z_{i1}, z_{i2}, z_{i3}]\) the set of discrete states for the association variable are:

<table>
<thead>
<tr>
<th>State</th>
<th>Association States</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 1</td>
<td>(x_{i1} - z_{i1}, x_{i2} - z_{i2}, x_{i3} - z_{i3})</td>
</tr>
<tr>
<td>State 2</td>
<td>(x_{i1} - z_{i2}, x_{i2} - z_{i1}, x_{i3} - z_{i3})</td>
</tr>
<tr>
<td>State 3</td>
<td>(x_{i1} - z_{i3}, x_{i2} - z_{i1}, x_{i3} - z_{i2})</td>
</tr>
<tr>
<td>State 4</td>
<td>(x_{i1} - z_{i2}, x_{i2} - z_{i3}, x_{i3} - z_{i1})</td>
</tr>
<tr>
<td>State 5</td>
<td>(x_{i1} - z_{i3}, x_{i2} - z_{i1}, x_{i3} - z_{i2})</td>
</tr>
<tr>
<td>State 6</td>
<td>(x_{i1} - z_{i3}, x_{i2} - z_{i2}, x_{i3} - z_{i1})</td>
</tr>
</tbody>
</table>

For state 1, target \(x_{i1}\) is associated with measurement \(z_{i1}\) AND target \(x_{i2}\) is associated with measurement \(z_{i2}\) AND target \(x_{i3}\) is associated with measurement \(z_{i3}\). The remaining states...
result from holding the order of the targets fixed and permuting the order of the measurements.

Note that the number of sensor measurements is not necessarily restricted to the number of targets within the sensor range to allow for false detections and missed detections.

All sensor nodes that share coverage of one or more targets (that is, have a predicted target location that is within their sensing range) are dependent and are connected with an undirected edge in the graphical model. The potential functions associated with an edge represent the strength of the relationship between states of the node RVs connected by an edge. They represent the joint likelihoods of particular combinations of measurement-target associations for each of the linked nodes. The specific details on how the potential functions are determined can be found in (Chen, Wainwright et al. 2003).

After construction of the basic graphical model, the next step is to determine the potential (compatibility) functions associated with each edge. If no target is shared by more than two sensors, then it is straightforward to determine the pair-wise node potential functions. However, if a target is shared by three or more sensors then the potentials are functions of all the nodes (the clique of nodes) which share the target and determination of the potential functions is more involved. Conversion of the general MRF to a pair-wise MRF considerably simplifies the description and analyses of message-passing (e.g. belief propagation) inference algorithms. Weiss (2000) notes that “The main advantage of the pairwise Markov net formulation is that it enables us to write the message-passing scheme in terms of matrix and vector operations, and this makes the subsequent analysis simpler.” In a pairwise MRF the maximal cliques are pairs of nodes.

Yedidia, Freeman et al. (2000) note that any graphical model can be converted to pair-wise MRF through the suitable grouping of nodes and the addition of compound auxiliary nodes to represent these groupings. Weiss (2000) presents a procedure for converting a general Bayesian Network into a pair-wise MRF. This conversion procedure maintains loops (if the Bayesian Network has loops then so does MRF), the update rules in the
MRF reduce to the messages in the original Bayesian Network and every message in the Markov network can be identified with a message in the Bayesian Network. Chen, Wainwright et al. (2003) describe a procedure for adding auxiliary random variables to convert the general MRF (with clique sizes of 3 nodes or greater) formed when targets are shared by three or more sensors into a pair-wise MRF. Their procedure has informed this work.

4.4 Network Topologies and Architectures

A key aspect of this research is to investigate the interrelationships between the physical, communications and inference and information fusion architectures on the performance of the inference/information fusion algorithms. Hence, algorithm performance was investigated for a number of inference and fusion architectures including fully distributed, locally distributed, centralized and locally centralized and for a spectrum of physical and communications architectures (e.g. how the nodes are clustered, if and how a master node in the cluster is selected, the number of nodes in a cluster). Figure 9 illustrates the various alternative network architectures. For a fully centralized architecture (Figure 9 upper left), all inference and fusion processing is performed at a central processing site. For a fully distributed architecture (Figure 9 upper right), the inference and information fusion processing is distributed among all nodes in the network. For a locally centralized architecture (Figure 9 lower left), a master node (clusterhead) is selected from among the nodes in a cluster and the processing of the inference and information fusion algorithm is performed at the clusterhead with sensor data supplied by the other nodes in the cluster. For a locally distributed architecture (Figure 9 lower right), the inference and information fusion processing is distributed locally among the physical sensor nodes in a cluster.
4.4.1 Localization and Clustering

The grouping of sensor nodes into clusters has been investigated for purposes of energy savings, increasing network life, reducing communication latencies and improving scalability. In most clustering schemes the sensor nodes communicate information only to cluster heads and the cluster heads communicate the aggregated information to the information sink.

For wireless sensor networks, it is desirable to have clustering algorithms that use only local information to organize the sensor nodes into clusters. Furthermore, for large wireless sensor networks, where energy conservation is of primary importance, it is critical to utilize a clustering procedure that is fast and that minimizes communications.
It should be noted that clustering can occur at multiple levels in a hierarchical fashion such that at higher levels you have clusters of clusters. In this work, however, we will be considering only one level of clustering.

We have an additional reason to consider clustering. Clustering is a logical mechanism for localization of the inference and information fusion calculations. The general concept of clustering is for the inference and fusion calculation to utilize (sensor) information that is local to the nodes in the cluster. It is anticipated that the localization of the calculation will result in a reduction in the communication and energy requirements and a reduction in communication latencies. In certain cases, however, localization and clustering may result in a loss of information with a resultant increase in the inference and fusion error rate. The goal then is to develop localization procedures with a favorable tradeoff between the (reduction in the) quality of the inference and information fusion results and the reduction in communication costs and latencies.

For the MTMS data association problem, the loss of information resulting from clustering is particularly severe for high target densities. Figure 10 shows the pairwise Markov Random Field graphical model for a 10x10 sensor grid with an average target density of three targets per sensor (each sensor covers on average three targets). Sensor nodes (shown as circles) connected by an edge share coverage of one or more targets and only these two nodes cover the targets. Sensor nodes with an edge connected to an auxiliary node (shown as a square) also share coverage of a target. However, this target is covered by three or greater sensors. The edges connecting nodes represent dependencies and allow for the joint association of node sensor measurements with a target. As seen in Figure 10 there is a dense pattern of node dependencies with nodes being connected to multiple neighbors with edges.

Clustering involves selecting subsets of nodes in network, developing graphical models over the nodes in the each cluster, and performing inference and information fusion within each cluster rather than over the entire network model. In developing the local
cluster graphical models, the treatment of the inter-cluster dependencies (edges that connect nodes in different clusters) is an important consideration.

Several novel techniques have been developed for treating inter-cluster dependencies. The simplest approach is to simply ignore this dependency (remove the edge). This is similar to the “arc removal” technique developed by van Engelen (1997) for simplifying loopy Bayesian Belief Networks (cutting loops such that the resultant network is singly connected). Removing an edge between two nodes removes the direct dependency between the nodes. For the MTMS data association problem, where an edge connecting
two nodes represents shared coverage of a target, edge removal results in each node associating its measurement set with the targets it covers without considering the joint likelihood represented by the potential functions associated with the edge.

Removing all edges interconnecting two disjoint sets of nodes results in non-overlapping clusters. The nodes in each cluster are rendered independent of the nodes in all other clusters. Figure 11 presents an example of clustering with edge removal for the same target configuration as in Figure 10.

![Figure 11 Pairwise MRFs for Node Clusters Created Using Edge Elimination Technique](image)

In Figure 11, edges shown by dashed red lines have been removed resulting in non-overlapping node clusters (indicated by different color nodes and edges). For example removing the following set of edges [96-97, 86-87, 96-87, 86-97, 79-89, 80-90, 79-90,
creates a cluster of nodes [77, 78, 87, 88, 97, 98, 89, 90, 99, 100] that are independent of all other nodes in the network. After clustering and edge removal a pairwise MRF graphical model is constructed for each cluster.

The second method that was investigated for treating inter-cluster dependencies involves the formation of overlapping node clusters. This technique was inspired by the method of cut set conditioning (Pearl 1988; Horvitz, Suermondt et al. 1989) developed to simplify loopy Bayesian Networks. For this method some nodes may be a member of more than one cluster. For a specific cluster the set of nodes that is contained in this cluster and in any other cluster is called a cut set or separation set. Nodes in the cut set have edges that connect to nodes contained only within the specified cluster and also have edges that connect to nodes in other clusters (and possibly to other cut set nodes). The cut set nodes have the property that when the value for the nodes in the cut set are instantiated (a specific state value for each node in the cut set is fixed), then the cluster is rendered conditionally independent of all nodes in all other clusters.

Alternatively, if the cut set nodes are reproduced (cloned) in each cluster and the clusters are separated by removing edges that connect the cut set nodes with nodes in any other cluster, then the clusters become independent of other clusters. For example, consider the simple network graphical model shown in Figure 12a.

Assume that our clustering algorithm organizes the network into two overlapping clusters as shown in Figure 12b with the cut set nodes shown by hatch filled circles. The cut set cloning methods then creates the two clusters shown in Figures 12c and 12d with the cut set nodes reproduced in each cluster.

For each cluster, we perform the inference/fusion calculation and inspect the maximum a posteriori (MAP) association state for each cut set node for each cluster calculation. If the MAP state for the cut set node is the same for all cluster calculations, then this state is selected for the node. However, if the MAP association state for the node is different for
different cluster calculations, differences are resolved by combining (multiplying) the marginals for each state for the cut set node from each cluster calculation and then choosing the state that has the greatest product.

In this work, clustering is performed for each localization and tracking cycle as shown in Figure 8. However, this may be wasteful of resources since clustering requires both additional communications and computation. Furthermore, the additional communications and computations associated with cluster formation will add to the time required to complete the localization and tracking cycle. Depending on the specific context of the situation (for example, if target movement during a cycle time interval is small relative to the sensing range of the sensors) it may be possible and energy efficient to avoid reclustering each cycle and to perform clustering after a number of cycles have
occurred. As the time increases since the last clustering and the target distribution changes within the sensor network, the existing clustering arrangement may become suboptimal but still be sufficiently representative to provide an acceptable level of performance for the network. Judgments regarding the acceptable level of performance and number of cycles between reclustering are application specific and will need to be made on a case-by-case basis; balancing inference and fusion quality versus the costs for more frequent clustering.

4.5 Inference Methods

The inference methods to be evaluated in this study are advanced approximation methods based on variational approximations to the Gibb’s free energy. As noted in previous chapters, Pearl’s belief propagation algorithm is an instance of the Bethe free energy approximation. The work by Chen, Wainwright et al. (2003) shows that application of a variant of Pearl’s BP algorithm works well for the multitarget multisensor (MTMS) data association problem.

4.6 Metrics

The principal output variables/performance metrics to be considered will include:

- Inference/Data Fusion Quality
  - Association error rate
- Communications (Energy) Costs
  - Number Word-Hops
- Communications Latencies
  - Number sequential hops

In most communications routing protocols, the optimal paths are selected based on minimizing hop count or communications delay (Salhieh and Schwiebert 2004).
The appendix provides a detailed breakdown for the process used to estimate the communication costs and communication latencies.

In addition to the principal performance metrics listed above, additional simulation results of interest include:

- Convergence/non-convergence of the inference/information fusion algorithm
- Number of iterations to convergence
- Scalability
- Robustness (single points of failure, non-uniformity of energy utilization)

4.7 Specific Research Plan

This research focuses on assessing advanced algorithm performance under the four diverse inference and information fusion architectures discussed above.

4.7.1 Simulation Assumptions

The assumptions that will be made regarding the network include:

- To minimize communications energy costs, all communications occur between any two points by multi-hopping over the shortest distance (Manhattan distance)
- For each hop the shortest hop distance is chosen – for a 2D square rectangular sensor grid this corresponds to a nodes nearest neighbors in the “x” or “y” coordinate directions
- All nodes in cluster are within $n$ communications hops from all other nodes in the cluster (where $n$ is a simulation parameter)
- Cluster formation has occurred
- Communication links have been established among nodes in a cluster
- All nodes know their own location and the locations of all other nodes in the cluster
- Communications are by wireless radio (or acoustic) transmissions
• All sensor signals are of a single type (modality) within a cluster (e.g. acoustic amplitude readings),
• Sensor noise is modeled by independent Gaussian distributions
• The environment will be represented by a two-dimensional world grid with sensors uniformly distributed in this world grid.

4.7.2 Test Variables

The principal independent (test) variables will include:

• Network Architecture – Four diverse architectures were investigated including:
  o Centralized
  o Fully distributed
  o Locally centralized
  o Locally distributed
• Number targets - Target densities in the range one to three targets per sensor were simulated
• Number of nodes in a cluster - For localized architectures cluster sizes (maximum widths) ranging from two to ten nodes were investigated
• Model Simplification Technique - For localized architectures two distinct techniques were developed and assessed for treating inter-cluster dependencies
Chapter 5  Simulation Results

This chapter presents the results of the simulation runs. As discussed in chapter 4, the three principal metrics that are being investigated are inference and data fusion quality (quantified by the percent of erroneous target-measurements associations), the amount of communications required (as a surrogate for energy use) and the latencies in the overall process.

5.1 Simulation Parameters

The results are presented for the four network architectures investigated: i) centralized, ii) fully distributed, iii) locally centralized, and iv) locally distributed. Results are presented for three average target densities (1, 2 and 3 targets per sensor). For each of the two architectures where the inference and data fusion is localized by cluster formation, results are presented for the two methods used to handle inter-cluster dependencies i) edge elimination and ii) cut set cloning. In addition, for the latter two architectures, calculations were performed to assess the impact of cluster size on performance. Table 2 presents a summary of the parameters for the simulation runs.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20x20 square 2d grid network of sensor nodes (400 sensors)</td>
<td></td>
</tr>
<tr>
<td>Ave Target Densities (TD) of 1, 2 and 3 targets covered per sensor (~ 134, 268, &amp; 403 targets)</td>
<td></td>
</tr>
<tr>
<td>Sensor node spacing = 5 (units)</td>
<td></td>
</tr>
<tr>
<td>Sensor coverage range = 5 (units)</td>
<td></td>
</tr>
<tr>
<td>Measurement variance $\lambda^2 = 2.5$ ($\lambda = 1.58$ units)</td>
<td></td>
</tr>
<tr>
<td>Prediction variance $\sigma^2 = 5.0$ ($\sigma = 2.24$ units)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Simulation Parameters

Figure 13 illustrates the sensor placement, sensor coverage areas, the target prediction uncertainty and sensor measurement uncertainty.
5.2 Association Error Rates

Figures 14, 15 and 16 present the association error results for average target densities of three, two and one targets per sensor, respectively. The x-axis is the parameter used to determine the maximum dimension of a cluster and is the maximum distance between any two nodes in a cluster. This parameter can be taken as the maximum cluster width. The number of nodes in a cluster will always be less than, or equal to, the square of this parameter.
Figure 14 Association Error for a Target Density of 3 Targets per Sensors

Figure 15 Association Error for a Target Density of 2 Targets per Sensor
Figure 16 Association Error for a Target Density of 1 Target per Sensor

The upper dashed line in these figures shows the association error if all edges were removed from the model. This represents the case where each sensor node determines the association of its measurements with targets in its coverage area based solely on its own set of measurements and independent from the measurements of its neighbors. Hence, this would represent the results without joint fusion of the measurement information.

The bottom solid line represents the results for joint inference and fusion performed on a model that considers the entire sensor network without localization and clustering and with all edges in the graphical model preserved. This result represents the best solution to be obtained using belief propagation on a probabilistic graphical model. Prior research has shown that when belief propagation converges, the results are generally a good approximation to the exact solution (Frey and MacKay 1998; Murphy, Weiss et al. 1999). The results presented for this case would result either from the inference and information fusion algorithm applied to the centralized processing architecture or to a fully distributed architecture.
The simulation results shown with “x” and “o” symbols are for the localized architectures (locally centralized or locally distributed) using the edge elimination and cut set cloning model simplification techniques to handle inter-cluster dependencies, respectively. It should be noted that the results indicated by each of the symbols represents the mean of 50 trials, where each trial represents a different placement of the “predicted” target locations.

5.3 Discussion of Association Error Rates

Comparing Figures 14, 15 and 16, it can be observed that as the target density decreases the association error decreases for all models used. This is easily understood since at the low target density (target density = 1) each sensor on average is covering one target and generating one measurement. Hence, it is a trivial calculation to associate this single measurement with the sole target in the coverage area. As the target density increases the number of possible association combinations increases as $n_!$ (where $n_!$ is the number of targets covered by a sensor) and the association error rate increases for all models.

Consider the upper dashed line (no data fusion) and the lower solid line (fusion over a model spanning the entire sensor network) in these figures. The difference between these two results represents the benefits of overlapping sensor coverage and information fusion in reducing the association error rate.

In general, the techniques involved in clustering and localizing the algorithm results in some loss of information. The extent and impact of this information loss depends on the manner in which the inter-cluster dependencies are handled during the localization and clustering process. From the standpoint of inference and data fusion solution quality, we would like a model simplification approach that minimizes the incremental error added as a result of clustering and localization.

Comparing the results for edge elimination with that of cut set cloning in Figures 14, 15 and 16, one observes that cut set cloning always results in a lower error rate than edge
elimination. These results are not surprising since whereas the edge elimination technique completely ignores inter-cluster dependencies, the cut set cloning approach treats these dependencies in an approximate manner resulting in improved performance.

One also observes from these figures that there is a decrease in error rate (for both approaches) with increasing cluster size (width). It is also evident that the rate of decrease, decreases with increasing cluster size. The decrease in error rate with increasing cluster size is explained by noting that as the cluster size increases the number of inter-cluster dependencies (edges connecting nodes in different clusters) decreases. Hence, the number of edges that are eliminated or treated approximately also decreases. The decreasing rate of decrease in the error rate with increasing cluster size can provide guidance to designers in choosing maximum clusters to sizes to “minimize” the association error. For the 20x20 grid we have used a good (if not optimal) maximum cluster size which appears to fall in the range of 6 to 8 (for the Comm Dist parameter used in cluster formation). Also note that for our 20x20 grid, clusters with widths greater than 10 nodes would force formation of other clusters in the network with widths less than 10 nodes. Hence, setting the maximum cluster “width” greater than the half-width of the sensor network should cause the error rate to increase.

5.4 Communications (Energy) Costs

The second metric used to measure the performance during the simulation runs was the number of communications that are required. The specific quantitative parameter that was evaluated was the total number of word-hops (see below) and the distribution of word-hops among the nodes in the network. Based on the discussion presented in Chapter 2 regarding the power requirements for transmitting messages, it appears that multi-hopping messages across short distances is generally preferable to long range single hop transmissions.

Consequently, we have assumed that the minimum energy requirements for communications can be realized by multi-hopping from nearest neighbor to nearest neighbor in the network. For our regular 2d grid arrangement this corresponds to a single
hop distance equal to the node grid spacing in the x or y direction. For multi-hopping from a given node in the network to any other node or to the information sink, the shortest distance is the Manhattan distance and the required number of hops is given by the Manhattan distance divided by the node spacing.

The length of each communication message (in bits) is determined by the type of information to be transmitted (real, integer, etc) and the required precision for this information. This information is highly application specific and we choose to measure the communication length by the number of words transmitted, where a word is a discrete piece of information such as the “x” or “y” Cartesian location of a sensor measurement (a real number) or a node ID (real, integer, or character string). Since the number of bits required to represent each of these discrete pieces of information may differ, the number of bits communicated would be equal to the average word length (in bits) times the number of words transmitted (plus the message overhead for coding, routing, error checking, etc).

Finally our communication metric is the number of Word-Hops, defined as the length of a message (number of discrete pieces of information transmitted) times the number of hops between message-initiating node and final message-receiving node (or sink).

Table 3 presents the total communication requirements for each architecture, broken down by task. The seven tasks shown are measurement communication, cluster formation, cluster head selection, inference data fusion, estimated target position calculation, inference fusion results to sink, and tracking position to sensors. The appendix provides a detailed description of how the communications requirements were estimated by task for each of the processing architectures. Note that the results are shown for the localized calculation for a cluster maximum “width” of six nodes (COMM_DIST = 6). Note also that the communications requirements for cluster formation and cluster head selection were not evaluated. It should be noted that the clustering algorithm that was employed for this work was ad-hoc and suboptimal. Consequently, it was not possible to estimate realistic communication requirements for these tasks.
Measurement communication reflects the transmission of each individual node sensor measurements to the processing site for inference and fusion. For a centralized architecture, this means transmitting the sensor measurements to the central processing site (sink). For the locally centralized processing architecture, this involves transmitting the sensor measurements to each cluster head for all the clusters in which the node is a member. For fully distributed processing, this involves each node sharing coverage of a target to send all its measurements to all other nodes covering the same target. For locally distributed processing, this involves each node sharing coverage of a target to send all its measurements to all other nodes covering the same target that are within the same cluster(s).

Inference and fusion communications reflect the message-passing between physical nodes associated with the distributed belief propagation algorithm processing. These communications are only required for those processing architectures with distributed processing (fully distributed and locally distributed).

Once each sensor measurement is associated with a target, the next step in the process is to produce updated estimates for each target position. In general, multiple sensors will generate a measurement for a specific target (because of sensor coverage overlap). Consequently, these measurements must be transmitted from the processing site where the inference/fusion was performed for each sensor to a common site for aggregation into a single estimate for target position.

After the updated target positions have been determined, they need to be communicated to the sink for ultimate transmission to the information consumer. They also need to be communicated to the site where the tracking algorithm is processed in order to develop the future (next timestep) prediction for target location.

Finally, the predicted target positions need to be communicated to the sensor nodes that cover the region where the targets are expected in the next measurement cycle.
<table>
<thead>
<tr>
<th>TASK</th>
<th>Centralized</th>
<th>Fully Distributed</th>
<th>Locally Centralized</th>
<th>Locally Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(Com_Dist = 6)</td>
<td>Non OL/OL</td>
<td></td>
</tr>
<tr>
<td>Measurement Communications</td>
<td>7.2e4</td>
<td>1.1e4</td>
<td>1.7e4/2.2e4</td>
<td>1.1e4</td>
</tr>
<tr>
<td>(Hops to sink)</td>
<td></td>
<td>(Hops to covering nodes)</td>
<td>(Hops to cluster head)</td>
<td>(Hops to covering nodes)</td>
</tr>
<tr>
<td>Cluster formation</td>
<td>NA</td>
<td>NA</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>Cluster head selection</td>
<td>NA</td>
<td>NA</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>Inference/data fusion</td>
<td>NA</td>
<td>3.2e6</td>
<td>NA</td>
<td>1.2e6</td>
</tr>
<tr>
<td>Estimated target position calculation</td>
<td>NA</td>
<td>3.6e3</td>
<td>7.5e3/0</td>
<td>3.6e3</td>
</tr>
<tr>
<td>Inference/ fusion results to sink</td>
<td>NA</td>
<td>2.4e4</td>
<td>2.4e4</td>
<td>2.4e4</td>
</tr>
<tr>
<td>Tracking predictions to sensors</td>
<td>7.2e4</td>
<td>7.3e3</td>
<td>1.7e4</td>
<td>7.3e3</td>
</tr>
<tr>
<td>TOTALS</td>
<td>1.4e5</td>
<td>3.3e6</td>
<td>6.6e4/6.3e4</td>
<td>1.3e6</td>
</tr>
</tbody>
</table>

Table 3 Communication Requirements by Task for Each Architecture
(NA –not applicable, NC – not calculated, OL – overlapping clusters)

As can be seen in Table 3, the total communications requirements are greatest for the distributed processing architecture, and the task that dominates the communication requirements for these architectures is the distributed inference/fusion processing.

Overall, the locally centralized architecture exhibits the lowest communication requirements of all the architectures.
Although the total communications requirements shown in Table 3 are important, an equally important consideration is the manner in which the communications are distributed among the nodes. Nodes with a communications burden well above the average will deplete their energy reserves most rapidly and will fail most rapidly leading to network degradation. Figure 17 shows the peak node and the average node communication requirements for the four architectures. Note that the peak node communications are highest for the centralized processing node. The nodes with these high peak values are those adjacent to the sink. These nodes must handle the communications traffic (sensor measurements) to the sink from all other sensor nodes in the network and the return traffic (target predictions) from the sink to the sensor nodes. Peak node communications are lower for all other architectures for two reasons. Firstly, instead of transmitting the individual sensor measurements to the sink, these architectures fuse the sensor measurements within the network and transmit the fused results (estimated target position) to the sink. This results in a reduction in the communications traffic by about two thirds for our example. Secondly, the tracking algorithm is also processed within the network. Consequently, the predicted target positions are generated locally and not at a central processing site.

![Figure 17 Distribution of the Communications for the Four Processing Architectures](image-url)
The average node communications load is dominated by the inference and fusion task for the distributed processing architectures (locally and fully distributed). The locally centralized architecture has both the lowest peak and lowest average communication requirements since this architecture has neither the communications load of transmitting sensor measurements to the sink nor the load imposed by distributed inference and fusion processing.

5.5 Communications Latencies

Communications latencies were counted as the number of sequential communications performed during a measurement cycle. Table 4 shows a breakdown of the sequential communications by task for the four processing architectures. The total communication latency is then the sum for all tasks. Note, however, that the last two tasks can be performed in parallel; hence only the maximum of either of these tasks should be added to the other tasks. The appendix provides a detailed description of how the communications latencies were estimated by task for each of the processing architectures.

The tasks that require that a communications be hopped from a node in the network to the sink require a substantial amount of time. These tasks include transmitting the sensor measurements to the sink (and then transmitting the predicted target locations from the sink back to the sensor nodes) for the centralized architectures and for transmitting the fused results (estimated target positions) from the nodes to the sink for the other architectures. For a sink location on the edge of the sensor network the maximum number of hops is approximately twice the width of the network following a Manhattan path.

The other major contribution to communications latencies are those associated with distributed inference and fusion processing. Although the communications distances are short (in our example a maximum of two hops), these communications must be repeated for each iteration while processing the inference and fusion algorithm.
<table>
<thead>
<tr>
<th>TASK</th>
<th>Centralized</th>
<th>Fully Distributed</th>
<th>Locally Centralized Com Dist = 6</th>
<th>Locally Distributed Com Dist = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement Communications</td>
<td>40</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Cluster formation</td>
<td>NA</td>
<td>NA</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>Cluster head selection</td>
<td>NA</td>
<td>NA</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>Inference/data fusion</td>
<td>NA</td>
<td>72</td>
<td>NA</td>
<td>32</td>
</tr>
<tr>
<td>Estimated target position</td>
<td>NA</td>
<td>2</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>calculation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inference/fusion results to sink</td>
<td>NA</td>
<td>max ( 40 or</td>
<td>max ( 40 or</td>
<td>max ( 40 or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7)</td>
<td></td>
<td>7)</td>
</tr>
<tr>
<td>Tracking predictions to sensors</td>
<td>40</td>
<td>4</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>TOTALS</td>
<td>80</td>
<td>116</td>
<td>62</td>
<td>86</td>
</tr>
</tbody>
</table>

Table 4 Communication Latencies by Task for Each Architecture

Note: The last two tasks can be performed in parallel.
Hence, only the maximum of either of these tasks should be added to the total.

As for the average and peak communication requirements, the communications latencies are a minimum for the locally centralized architecture since this architecture does not involve distributed processing of the inference/fusion algorithm nor is there the need for transmitting the predicted target positions from the sink to the sensor nodes (as for the centralized processing architecture).
Chapter 6 Contributions, Conclusions and Future Research

This chapter summarizes the contributions of this dissertation, highlights several of the main findings and conclusions of the research and suggests future directions for research.

Chapter 1 identifies the many areas of application for large scale networks of inexpensive wireless sensor nodes. Chapter 1 discusses the challenges associated with wireless sensor network design and indicates that development of inference and information fusion techniques that are scalable to large sensor networks that can be applied locally and that minimize the consumption of scarce network resources (energy supplies and communication bandwidth) is one of the most important research needs.

6.1 Contributions

The major contributions of this research are the development of inference and information fusion techniques that are scalable to large sensor networks, that can be applied locally, and that minimize the consumption of scarce network resources (energy supplies and communication bandwidth).

The specific contributions of this dissertation include:

- Dynamic Model Generation
- Procedures for Localizing Computation
- Methods for Model Simplification
- Identification of Important Performance Metrics
  - Inference Quality
  - Communications Costs (Energy Usage)
  - Communications Latencies
- Quantitative Results for the Performance Metrics for Four Network Architectures
These specific contributions are described in more detail in the following sections.

6.1.1 Dynamic Model Generation

For each measurement cycle for the multtarget-multisensor localization and tracking problem, a new graphical model (pairwise Markov Random Field) is generated based on the current problem conditions (predicted target locations). The dependencies (existence of edges) exhibited in the model guide the localization of the calculation (clustering) resulting in different clusters for each cycle. This process naturally helps to balance resource expenditures in the sensor nodes by temporally distributing the computation and communications loads on the sensor nodes.

6.1.2 Procedures for Localizing Computation

For the localized architectures (locally centralized and locally distributed), procedures were developed for cluster generation that utilize local information. For the localized architectures, the inference and fusion processing is performed within a cluster using information (sensor measurements) that are local to the cluster. This processing is either locally centralized in the cluster head nodes or locally distributed among the sensor nodes within the cluster. Options for trading off inference quality for energy use/communications latency were explored including the impacts of different cluster sizes and methods for handling inter-cluster dependencies (see Model Simplification section below).

6.1.3 Methods for Model Simplification

For the localized architectures, several methods were implemented for treating the inter-cluster dependencies. These included the edge elimination method and the method we have called “cut set cloning”. The edge elimination method is computationally simple in that it ignores inter-cluster dependencies, whereas the somewhat more computationally intensive “cut set cloning” technique treats these dependencies in an approximate manner.
6.1.4 Performance Metrics

To assess the performance of the various approaches we have implemented for application of the inference and information fusion algorithm to the various network architectures, this work has identified and developed quantitative metrics for the important wireless sensor network parameters. The three main performance metrics and the quantitative parameter that was measured for each are:

- Inference/Information Fusion Quality
  - Association error rate
- Communications (Energy) Costs
  - Number Word-Hops
- Communications Latencies
  - Number sequential hops

6.1.5 Quantitative Results for Metrics

Quantitative results for these metrics were developed for four network architectures:

- Centralized
- Fully Distributed
- Locally Centralized
- Locally Distributed

In addition, for the localized architectures, the metrics were quantified for different cluster sizes and for the two model simplification techniques that were developed. This work provides guidance for applying message-passing belief propagation algorithms for inference and information fusion in wireless sensor network applications.

6.2 Conclusions

This section summarizes the main insights and conclusions developed from this work.
6.2.1 Localization

The results of this work suggest that localization of the inference and fusion processing can reduce the overall communications and energy requirements of the network with a modest decrease in the quality of the inference and fusion results.

6.2.2 Distributed Processing

The results of this work strongly suggest that distributed processing (either fully distributed across the network or locally distributed in a cluster) of the inference and fusion algorithms is wasteful of network resources and a locally centralized architecture is preferred.

6.2.3 Belief Propagation and Advanced Belief Propagation

In performing this work it was found that the standard belief propagation (BP) algorithm is fast and appears to always converge (or at least be near convergence) for a modest number of iterations. (The convergence criteria used was the maximum change in any sensor node state marginal probability.) When BP hadn’t converged, it was near convergence (the max marginal state wasn’t changing). The literature indicates that when BP converges it will have converged to good approximation of the exact solution. Consequently, although advanced BP algorithms were implemented, it was not found necessary to use these because of the good performance of standard BP.

For applications where standard BP is found to suffer convergence problems, the advanced BP algorithms, discussed in Chapter 3, may be utilized. However, use of these algorithms may potentially add significantly to the computation and/or communications costs associated with the inference and information fusion calculations.
6.2.4 Scalability

The belief propagation (BP) algorithm applied to the data association problem for localization and tracking has a complexity $O(n!)^2N$ compared to exact solution techniques which have complexity $O(n!)^N$ (Chen, Wainwright et al. 2003) where:

- $N = \text{number of sensors in the network}$
- $n = \text{number of targets in range of each sensor}$.

Hence, for this problem the BP algorithm scales approximately linearly with the size of the network. If the computation is distributed in some fashion, then the upper size of the network does not appear to be limited by the inference and information fusion processing. However, other constraints may limit the size of the network. For example, communication latencies in hopping network results to the information consumer.

Noting that the complexity of the algorithm scales as $(n!)^2$, high target densities can lead to both time and space computational problems. Experience in performing this work suggests that average target densities exceeding 5-8 targets per sensor can be troublesome.

6.3 Future Research

A number of areas for future research have been identified. These are discussed below.

6.3.1 Optimal Clustering Algorithm

Probably, the most significant area of needed research involves development of fast localized algorithms that minimize communications and that consider the impacts on solution quality of the clustering process.

The ad-hoc algorithms that were used in this research are suboptimal. They don’t minimize communications nor do they minimize information loss during the clustering
The performance would almost certainly be improved with a better clustering algorithm.

6.3.2 Apply Techniques to More Realistic Environments

The wireless sensor network and graphical model simulation was performed within the Matlab environment. However, as discussed in Chapter 4, using a high level simulation environment such as Matlab may abstract too far away from the actual physical system and hide critical interactions among the environment, sensor hardware and software, communications, and inference and fusion algorithms. In particular, the effects of limited node processor memory, timing and latency issues, communication issues (transmission/reception errors and interference, asynchronous communication) and extensive concurrent interactions with the physical environment may not become evident. It is not clear whether a homogeneous network of low capability nodes (such as the Crossbow motes) have the computational capacity to process the required inference and fusion algorithms while at the same time controlling the communications and processing the sensor data. It may well be that a heterogeneous network, which includes a number of higher capability processing nodes, may be required. This would argue for a locally centralized architecture with these higher capability nodes acting as cluster heads. Consequently, it is suggested that these techniques be applied under more realistic environments including low level wireless sensor network simulation environment such as EMSim and on real wireless sensor network hardware.

6.3.3 Extend Problem Parameter Space

The simulations performed as part of this work employed a number a number of simplifying assumptions including fixed (non-mobile) sensor nodes, no missed or false detections, no sensor failures, and error free communications. Although it is believed that these assumptions do not impact the generality of the results it would be of interest to test these beliefs by extending the work such that these assumptions can be relaxed or removed.
Appendix – Communications Breakdown for Each Processing Architecture

This appendix describes how the communications requirements were estimated for performing the prediction, measurement, association and prediction steps for the multitarget-multisensor localization and tracking problem. These results are for one measurement cycle and are provided for each of the four processing architectures that were investigated.

For each of the architectures, the communication requirements are identified for each of the following processing tasks:

- Transmitting measurements to processing site(s)
- Cluster formation (not calculated)
- Cluster head selection (not calculated)
- Inference/data fusion processing
- Estimating target position
- Transmitting inference/fusion results to sink
- Transmitting tracking predictions to sensors

This breakdown is not unique. There may be other ways of organizing the communications which result in fewer (or greater) communications.

1 Global Assumptions

There are no missed or false detections. Hence, each sensor will generate one measurement for each target within its coverage range. All nodes know the communication pathway to the sink (that is, they know which neighboring node to hop information to in order for it to be routed to the sink).

2 Nomenclature

- $M_{ave}$ - Average number of measurement generated per sensor
- $N_{aux\_edge}$ – Total number of edges from auxiliary nodes to sensor nodes
• \( N_{\text{aux\_edge\_ave}} \) – Average number of edges from each auxiliary nodes to sensor nodes
• \( N_{\text{aux\_state}} \) – Average number of auxiliary node states
• \( N_{\text{cluster}} \) - The number of clusters in the model
• \( N_{\text{clust\_sens}} \) - Average number of clusters a sensor is in
• \( N_{\text{clust\_cutset}} \) - Average number of clusters each cut set node is in
• \( N_{\text{cut\_edge}} \) - Number of edges between nodes in different clusters that were removed
• \( N_{\text{cutset\_nodes}} \) - Number of cut set nodes
• \( N_{\text{edge}} \) - Number of edges in graphical model
• \( N_{\text{iter}} \) - The number of iterations to converge the inference algorithm
• \( N_{\text{sens}} \) - Number of sensors in network
• \( N_{\text{sen\_edge}} \) – Number of sensor node to sensor node edges
• \( N_{\text{sens\_state}} \) – Average number of sensor node states
• \( N_{\text{sens\_targ}} \) - Average number of sensors covering a target
• \( N_{\text{state\_cutset}} \) – Average number of cut set node states
• \( N_{\text{targ}} \) - Total number of targets in sensor field coverage area
• \( \text{NA} \) – Not applicable

3 Processing Architectures

3.1 Fully Distributed Processing

For the fully distributed processing architecture, the graphical model spans the entire sensor network and the inference and data fusion computation is distributed and performed locally at the sensor nodes. In an iterative manner each sensor combines its local evidence with messages from its neighboring sensor nodes with which it shares coverage of one or more targets and computes the association state with the maximum a posteriori probability (MAP).

Assumptions:

- Each sensor node knows the locations and IDs of all sensor nodes within its (radio) transmission range.
- Each sensor node knows the locations and IDs of all sensor nodes with which it shares coverage overlap.
- Each sensor knows the coverage range of all (neighboring) sensors
At the beginning of the measurement cycle, each sensor whose sensor range covers a predicted target location receives a message with the target ID and the predicted target position. It then generates a set of measurements (equal to the number of targets in its coverage range). For our pairwise MRF model, for each target covered by 3 or more sensors, an auxiliary variable is constructed with (auxiliary node) edges added from the auxiliary node to each of the associated sensor nodes. Processing for the auxiliary variable is assigned to one of the physical sensor nodes sharing coverage of the target. (For example, the sensor node closest to the predicted target position.) For any pair of sensors that share coverage of targets, all of which are covered by 3 or more sensors, then the direct edge between these sensors is removed. Messages are composed by each sensor with its current set of measurements and transmitted to each of its neighbors with which it shares coverage overlap with one or more targets. Using this information, each sensor constructs the compatibility matrix (potential functions) relating its association states with its neighbors states with which it is connected by a direct edge in the graphical model. Compatibility matrices are also constructed for the auxiliary nodes (by the physical sensor node processing the auxiliary node variable) that relate the auxiliary node variable states to the states of its associated sensor nodes.

An iterative process begins where each node transmits (and receives) a message to (from) its neighbors with which it shares an edge (both along direct edges connecting two sensor nodes and along the edges connecting the sensor node processing an auxiliary variable for a specific target and the other sensor nodes covering this target). The message length is equal to the number of states in the message receiving node (either the number of sensor node association states or number of auxiliary node states).

After receiving the messages, each node updates its local beliefs (marginal probabilities) and determines the MAP state. It then checks to see if its local beliefs have converged and continues with the iterative process. The iterative inference process is stopped when all sensor/auxiliary nodes have converged.
Once the iterative process has been completed, the association results can be used to generate updated estimates of the positions for each target (at the current measurement time) and the future predicted target position (next measurement time). A physical sensor node must be selected to perform this processing for each target. It could be selected based on choosing the closest physical node to the predicted target location (at the beginning of the measurement cycle). This is the same criteria used to select the physical sensor node to perform processing for the auxiliary variables associated with a target.

Messages are generated by all nodes that share coverage overlap of a target and are transmitted to this processing node. This node then combines the measurements generated by each covering sensor and associated with the target to produce an updated estimated target position (for example by computing the average of the $x$-coordinate values and average of the $y$-coordinate values of each associated measurement). Next, the node, using a tracking/prediction algorithm, computes the future predicted position of the target.

The sensor node which has computed the current estimated position and future “predicted” position of the target generates messages to all sensors that cover the future predicted position of the target giving the target ID and the $(x,y)$ coordinates of the future predicted position. It also generates a message to the sink giving the target ID and the $(x,y)$ coordinate for its current estimated position and possibly its future predicted position (since the sink has access to the same tracking/prediction algorithm as the sensor nodes, it may be beneficial in terms of communication savings to recalculate the future positions at the sink rather than to communicate them from the local source).

Table 5 provides a breakdown of the communication requirements for the fully distributed processing architecture.

3.2 Centralized Processing

All inference and data fusion computation is performed at a remote node (the centralized processing node). This node is assumed to have a large energy reserve and have a large processing capability.
Assumptions:

- Each sensor node knows the locations and IDs of all sensor nodes within its (radio) transmission range.

Each sensor node that observes one or more targets (and generates a measurement for each target) composes a message to be sent to the centralized processing node which includes the $x$ and $y$ coordinates for each measurement and the sensor node ID (or location). This message is multi-hopped from the sensor node to the centralized processing node.

The central processing node receives all messages, associates each sensor measurement with a target and generates an estimate of the current position for each target. The current target positions are input into the tracking and prediction algorithm to generate predicted positions for each target in the future. The current target positions and predicted future prediction are then sent to the information consumer.

The central processing node generates messages to all sensors that cover the future predicted position of the target giving the target ID and the $(x,y)$ coordinates of the future predicted position and transmits these messages back to appropriate sensor nodes.

The inference and data fusion processing performed by the centralized processing node is similar to that described for the fully distributed architecture. The major differences being that the centralized processing node internally constructs a graphical model containing (inference) nodes representing the physical sensor nodes for all the sensors in the network and their associated auxiliary variables and the entire iterative inference and data fusion processing for the network is performed at one physical location without the need for physical node to physical node communications.

Table 6 provides a breakdown of the communication requirements for the centralized processing architecture.
3.3 Locally Distributed Processing

In this processing architecture, local clusters of sensor nodes are formed and a sensor node is selected as the cluster head. For the locally distributed processing architecture, the graphical model spans only the nodes in the cluster and the inference and data fusion computation is distributed and performed locally at the sensor nodes in the cluster. In an iterative manner, each sensor combines its local evidence with messages from its neighboring sensor nodes with which it shares coverage of one or more targets and computes the association state with the maximum \textit{a posteriori} probability (MAP).

Assumptions:

- Each sensor node knows the locations and IDs of all sensor nodes within its (radio) transmission range.
- Each sensor node knows the locations and IDs of all sensor nodes with which it shares coverage overlap.
- Each sensor knows the coverage range of all (neighboring) sensors

At the beginning of the measurement cycle, each sensor whose sensor range covers a predicted target location receives a message with the target ID and the predicted target position. Based on this information, each node can determine which sensors it shares coverage of a target with and hence is connected with by an edge. Using this information and a known criterion for maximum cluster size and type of clustering to be performed (for example overlapping or non-overlapping clusters) cluster formation occurs. Once clusters formation has occurred then a cluster head is selected (for example, by choosing the sensor node whose location is closest to the centroid of the cluster).

Each sensor node then generates a set of measurements (equal to the number of targets in its coverage range). Each sensor then composes a message with its current set of measurements and transmits this message to each of its neighbors within the cluster with which it shares coverage overlap with one or more targets. The inference and fusion
processing is performed in a similar manner to that described for fully distributed processing and the target-measurement association results are transmitted from the cluster sensor nodes to the cluster head.

For cluster formation techniques that result in non-overlapping clusters, inference and data fusion processing is complete (edge elimination methods). However, for clustering methods that result in overlapping clusters, a further step is necessary. For these cases the same physical nodes may reside in multiple clusters. These “cut set” nodes may have different MAP assigned states and hence different measurement-to-target associations that need to be resolved. Since the calculations are performed locally, the cut set nodes have their association state results for each cluster calculation in which they are a member.

The MAP state assignments the cut set nodes are compared for each cluster calculation and if they all agree then the cut set node state assignment is set to this value. If they disagree then the marginals for the cut set node from each cluster calculation are combined (the marginals for each state are multiplied together) and the state assignment for the cut set node is selected as the state that has the maximum product.

Once the inference fusion and cut set resolution processes have been completed, the association results can be used to generate updated estimates of the positions for each target (at the current measurement time) and the future predicted target position (next measurement time) for targets covered by nodes in the cluster. A physical sensor node must be selected to perform this processing for each target. It could be selected based on choosing the closest physical node to the predicted target location (at the beginning of the measurement cycle) as in the fully distributed architecture. Alternatively, the cluster head can perform the estimation calculations. Assuming the latter is the case, messages are generated by all nodes that share coverage overlap of a target and are transmitted to the cluster head. The cluster head then combines the measurements generated by each covering sensor and associated with the target to produce an updated estimated target
position. Next, the cluster head, using a tracking/prediction algorithm, computes the future predicted position of the target.

The current target positions (and possibly predicted future prediction) are then sent to the sink. The cluster head node also sends the predicted target positions and target IDs to all sensor nodes (either inside or outside the current cluster) whose coverage area includes the predicted target position.

Note that even for non-overlapping clusters, one or more targets covered by a sensor in the cluster may also be covered by another sensor in a different cluster. (For example, when cluster formation occurs by edge removal the edge removed connects nodes in different clusters that observe the same target.) For these cases, the measurements from nodes in different clusters covering the same target must be combined to generate the updated target position (this calculation is assumed to be performed by the cluster head node nearest to the predicted target location).

Table 7 provides a breakdown of the communication requirements for the locally distributed processing architecture.

3.4 Locally Centralized Processing

In this processing architecture, local clusters of sensor nodes are formed and a sensor node is selected as the cluster head. Measurement information generated by the sensor nodes in the cluster are transmitted to the cluster head where the inference and data fusion processing is performed.

Assumptions:

- Each sensor node knows the locations and IDs of all sensor nodes within its (radio) transmission range.
- Each sensor node knows the locations and IDs of all sensor nodes with which it shares coverage overlap.
- Each sensor knows the coverage range of all (neighboring) sensors
At the beginning of the measurement cycle, each sensor whose sensor range covers a predicted target location receives a message with the target ID and the predicted target position. Based on this information, each node can determine which sensors it shares coverage of a target with and hence is connected with by an edge. Using this information and a known criterion for maximum cluster size and type of clustering to be performed (for example overlapping or non-overlapping clusters) cluster formation occurs. Once clusters formation has occurred then a cluster head is selected (for example, by choosing the sensor node whose location is closest to the centroid of the cluster).

Each sensor node then generates a set of measurements (equal to the number of targets in its coverage range). Each sensor then composes a message with its current set of measurements and transmits this message to head(s) of the cluster(s) within which it is located.

Each cluster head node receives all messages from nodes in its cluster, associates each cluster node sensor measurement with a target and generates an estimate of the current position for each target covered by one or more sensor nodes in the cluster. The current target positions are input into the tracking and prediction algorithm to generate predicted positions for each target in the future. The current target positions (and possibly predicted future prediction) are then sent to the sink. The cluster head node also sends the predicted target positions and target IDs to all sensor nodes (either inside or outside the current cluster) whose coverage area includes the predicted target position.

The inference and data fusion processing performed by the cluster head node is similar to that described for the fully distributed architecture. The major differences being that the cluster head node internally constructs a graphical model containing (inference) nodes representing the physical sensor nodes in the cluster and the associated auxiliary variables and the entire iterative inference and data fusion processing for the cluster is performed at one physical location within the cluster head without the need for physical node-to-physical node communications.
For cluster formation techniques that result in non-overlapping clusters, inference and data fusion processing is complete (edge elimination methods). However, for clustering methods that result in overlapping clusters, a further step is necessary. For these cases the same physical nodes may reside in multiple clusters. These cut set nodes may have different MAP assigned states and hence different measurement to target associations that need to be resolved. To resolve any differences, the cluster heads for each cluster containing the cut set node must transmit a message to a processing node (which could be one of the cluster head nodes or even the cut set node itself) with the MAP state number assignment (and the last iteration marginals for each cut set node state) for the cut set node for that cluster. The cluster head then resolves the cut set node association state as described above.

In addition, as described above, even for non-overlapping clusters, one or more targets covered by a sensor in the cluster may also be covered by another sensor in a different cluster. For these cases the measurements from nodes in different clusters covering the same target must be combined to generate the updated target position. (This calculation is assumed to be performed by cluster head node nearest to the predicted target location).

Table 8 provides a breakdown of the communication requirements for the locally centralized processing architecture.

4 Communication Latencies

Communications latencies were counted as the number of sequential communications performed during a measurement cycle. Table 9 shows a breakdown of the sequential communications by task for the four processing architectures. The total communication latency is then the sum for all tasks. Note, however, that the last two tasks can be performed in parallel; hence only the maximum of either of these tasks should be added to the other tasks.
| Measurement Communications | Total number of targets ($N_{targ}$) x 
Average number of sensors covering each target ($N_{sens\_targ}$) x 
Average number of sensors covering each target – 1 ($N_{sens\_targ} - 1$) x 
(Each sensor covering a target must send its set of measurements to all other sensors covering target) 
Average number of measurements per sensor ($M_{ave}$) x 
Message length (Target ID + x,y coordinates of measurement = 3) x 
Average number of hops from sensor covering target to neighboring sensor covering target (Min= 1, Max = 2, Use 1.5 as typical) |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster formation</td>
<td>NA</td>
</tr>
<tr>
<td>Cluster head selection</td>
<td>NA</td>
</tr>
</tbody>
</table>
| Inference/data fusion | Number of iterations during inference/data fusion ($N_{iter}$) x 
(Number of sensor node to sensor node edges in graphical model ($N_{sen\_edge}$) x 
Average number of sensor node states ($N_{sen\_state}$) x 
(Number of messages per edge = 2) + 
Average number of auxiliary node edges that connect to physically separate sensor nodes = 
(Number of sensor node to auxiliary node edges in graphical model ($N_{aux\_edge}$) x 
(fraction of aux node edges between physically separate nodes) 
($N_{aux\_edge\_ave} - 1$)/$N_{aux\_edge\_ave}$) x 
(Average number of sensor node states ) $N_{sen\_state}$) + 
Average number of auxiliary node states ($N_{aux\_state}$) x 
Average number of hops during inference (Min= 1, Max = 2, Use 1.5 as typical) |
| Estimated target position calculation | Total number of targets ($N_{targ}$) x 
Average number of sensors covering each target - 1 ($N_{sens\_targ} - 1$) x 
Messages length (Target ID + x,y coordinates of measurement associated with target = 3) x 
Average number of hops from sensors covering target to tracking node for target (Min= 1, Max = 2, Use 1.5 as typical) |
| Inference/fusion results to sink | Total number of targets ($N_{targ}$) x 
Messages length (Target ID + x,y coordinates of estimated target position = 3) x 
Average number of hops from local tracking node to sink 
($\sqrt{N_{sens}}$ = average number of hops from any sensor to sink located in or on edge of sensor field) |
| Tracking predictions to sensors | Total number of targets ($N_{targ}$) x 
Average number of sensors covering each target ($N_{sens\_targ}$) x 
Messages length (Target ID + x,y coordinates of predicted target location = 3) x 
Average number of hops from local tracking node to each sensor node covering target (Min = 1, Max = 3, Use 2 as typical) |

**Assumptions:**
1) Target maximum speed less than sensor spacing divided by inference cycle interval
2) Sink has same prediction algorithm as local sensors so sink can also calculate predicted target positions, hence no need to hop predictions to sink

Table 5 Communication Breakdown by Task for Fully Distributed Architecture
<table>
<thead>
<tr>
<th>Measurement Communications (Sensor Nodes to Sink)</th>
<th>Total number of sensors ($N_{sens}$) $\times$ Average number of measurements per sensor ($M_{meas}$) $\times$ Message length (Target ID + x,y coordinates of measurement = 3) $\times$ Average number of hops from sensor node to sink ($\sqrt{N_{sens}}$ = average number of hops from any sensor to sink located in or on edge of sensor field)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster formation</td>
<td>NA</td>
</tr>
<tr>
<td>Cluster head selection</td>
<td>NA</td>
</tr>
<tr>
<td>Inference/data fusion</td>
<td>NA</td>
</tr>
<tr>
<td>Estimated target position calculation</td>
<td>NA</td>
</tr>
<tr>
<td>Inference/fusion results to sink</td>
<td>NA</td>
</tr>
<tr>
<td>Tracking predictions to sensors</td>
<td>Same as Measurement Communications (assuming multi-hopping back from sink to sensor nodes)</td>
</tr>
</tbody>
</table>

Table 6 Communication Breakdown by Task for Centralized Architecture
### Table 7: Communication Breakdown by Task for Locally Distributed Architecture

<table>
<thead>
<tr>
<th>Task</th>
<th>Formulation</th>
</tr>
</thead>
</table>
| Measurement Communications                      | Total number of targets \((N_{\text{targ}})\) x 
Average number of sensors covering each target \((N_{\text{sens/targ}})\) x 
Average number of sensors covering each target - 1 \((N_{\text{sens/targ}} - 1)\) x 
(Each sensor covering a target must send its set of measurements to all other sensors covering target) 
Message length (Target ID + x,y coordinates of measurement = 3) 
Average number of hops from sensor covering target to neighboring sensor covering target \((\text{Min}=1, \text{Max}=2, \text{Use } 1.5 \text{ as typical})\) |
| Cluster formation                               | NA                                                                      |
| Cluster head selection                          | NA                                                                      |
| Inference/data fusion                           | Number of iterations during inference/data fusion \((N_{\text{iter}})\) x 
(Number of sensor node to sensor node edges in graphical model \((N_{\text{sen_edge}})\) x 
Average number of sensor node states \((N_{\text{sen_state}})\) x 
(Number of messages per edge = 2) + 
Average number of auxiliary node edges that connect to physically separate sensor nodes = 
(Number of sensor node to auxiliary node edges in graphical model \((N_{\text{aux_edge}})\) x 
(fraction of aux node edges between physically separate nodes) \((N_{\text{aux_edge_ave}} - 1)/N_{\text{aux_edge_ave}}\) x 
[Average number of sensor node states \((N_{\text{sen_state}})\) + 
Average number of auxiliary node states \((N_{\text{aux_state}})\)] x 
Average number of hops during inference \((\text{Min}=1, \text{Max}=2, \text{Use } 1.5 \text{ as typical})\) |
| Estimated target position calculation           | Total number of targets \((N_{\text{targ}})\) x 
Average number of sensors covering each target - 1 \((N_{\text{sens/targ}} - 1)\) x 
Message length (Target ID + x,y coordinates of measurement associated with target = 3) x 
Average number of hops from sensors covering target to tracking node for target \((\text{Min}=1, \text{Max}=2, \text{Use } 1.5 \text{ as typical})\) |
| Inference/fusion results to sink                | Total number of targets \((N_{\text{targ}})\) x 
Messages length (Target ID + x,y coordinates of estimated target position = 3) x 
Average number of hops from local tracking node to sink \((\sqrt{N_{\text{sen}}} = \text{average number of hops from any sensor to sink located in or on edge of sensor field})\) |
| Tracking predictions to sensors                 | Total number of targets \((N_{\text{targ}})\) x 
Average number of sensors covering each target \((N_{\text{sens/targ}})\) x 
Messages length (Target ID + x,y coordinates of predicted target location = 3) x 
Average number of hops from local tracking node to each sensor node covering target \((\text{Min}=1, \text{Max}=3, \text{Use } 2 \text{ as typical})\) |
| Assumptions:                                   | 3) Target maximum speed less than sensor spacing divided by inference cycle interval |
|                                                | 4) Sink has same prediction algorithm as local sensors so sink can also calculate predicted target positions, hence no need to hop predictions to sink |

108
### Measurement Communications

- Total number of targets \((N_{\text{targ}})\) \(\times\)
- Average number of sensors covering each target \((N_{\text{sens/targ}})\) \(\times\)
- Message length \((\text{Target ID } + \text{x,y coordinates of measurement} = 3)\)
- Average number of hops from each sensor node covering target to cluster head node \((\text{use } \sqrt{\text{cluster size as typical}})\) \(\times\)
- Average number of clusters a sensor is in \((N_{\text{clust/sens}})\)
  - 1 for non-overlapping clusters
  - > 1 for overlapping clusters

### Cluster formation

- -

### Cluster head selection

- -

### Inference/data fusion

- NA

### Estimated target position calculation

- Overlapping Clusters - 0
- Non-overlapping clusters:
  - Number edges between nodes in different clusters removed \((N_{\text{cut/edge}})\) \(\times\)
  - Messages length \((\text{Target ID } + \text{x,y coordinates of measurement associated with target} = 3)\) \(\times\)
  - Average number of hops between cluster head nodes \((\text{use } 2 \times \sqrt{\text{cluster size as typical}})\)

### Inference/fusion results to sink

- Total number of targets \((N_{\text{targ}})\) \(\times\)
- Messages length \((\text{Target ID } + \text{x,y coordinates of estimated target position} = 3)\) \(\times\)
- Average number of hops from cluster head node to sink
  \((\sqrt{N_{\text{sens}}} = \text{average number of hops from any sensor to sink located in or on edge of sensor field})\)

### Tracking predictions to sensors

- Total number of targets \((N_{\text{targ}})\) \(\times\)
- Average number of sensors covering each target \((N_{\text{sens/targ}})\) \(\times\)
- Messages length \((\text{Target ID } + \text{x,y coordinates of predicted target location} = 3)\) \(\times\)
- Average number of hops from cluster head node to each sensor node covering target \((\text{use } \sqrt{\text{cluster size as typical}})\)

**Assumptions:**
- Target maximum speed less than sensor spacing divided by inference cycle interval

<p>| Table 8 Communication Breakdown by Task for Locally Centralized Architecture |</p>
<table>
<thead>
<tr>
<th>TASK</th>
<th>Centralized</th>
<th>Fully Distributed</th>
<th>Locally Distributed Com_Dist = 6</th>
<th>Locally Distributed Com_Dist = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement</td>
<td>Max number of hops from sensor covering target to sink $2 \times \sqrt{N_{sens}}$</td>
<td>Max number of hops from sensor covering target to neighboring sensor covering target (Max = 2)</td>
<td>Max number of hops from sensor covering target to cluster head $\sqrt{N_{sens/cluster}}$</td>
<td>Max number of hops from sensor covering target to neighboring sensor covering target (Max = 2)</td>
</tr>
<tr>
<td>Communications</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster formation</td>
<td>NA</td>
<td>NA</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>Cluster head selection</td>
<td>NA</td>
<td>NA</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>Inference/data fusion</td>
<td>NA</td>
<td>Number of iterations $(N_{iter}) \times$ Max number of hops from sensor covering target to neighboring sensor covering target (Max = 2)</td>
<td>NA</td>
<td>Number of iterations $(N_{iter}) \times$ Max number of hops from sensor covering target to neighboring sensor covering target (Max = 2)</td>
</tr>
<tr>
<td>Estimated target position calculation</td>
<td>NA</td>
<td>Max number of hops from sensor covering target to neighboring sensor covering target (Max = 2)</td>
<td>Overlapping Clusters – 0 Non_Overlapping Clusters Max number of hops from cluster head to neighboring cluster head $2 \times \sqrt{N_{sens/cluster}}$</td>
<td>Max number of hops from sensor covering target to cluster head $\sqrt{N_{sens/cluster}}$</td>
</tr>
<tr>
<td>Inference/fusion results to sink</td>
<td>NA</td>
<td>Max number of hops from sensor covering target to sink $2 \times \sqrt{N_{sens}}$</td>
<td>Max number of hops from cluster head to sink $2 \times \sqrt{N_{sens}}$</td>
<td>Max number of hops from cluster head to sink $2 \times \sqrt{N_{sens}}$</td>
</tr>
<tr>
<td>Tracking predictions to sensors</td>
<td>Max number of hops from sink to sensor covering target $2 \times \sqrt{N_{sens}}$</td>
<td>$2 \times \sqrt{N_{sens}}$</td>
<td>Max number of hops from cluster head to sensor covering target $\sqrt{N_{sens/cluster}}$ + 2</td>
<td>Max number of hops from cluster head to sensor covering target $\sqrt{N_{sens/cluster}}$ + 2</td>
</tr>
</tbody>
</table>

Table 9 Sequential Communications for Each Architecture
Bibliography


Cozman, F. (2000). Generalizing Variable Elimination in Bayesian Networks. Workshop on Probabilistic Reasoning in Artificial Intelligence, Atibaia, Brazil


VITA
RICHARD R. SHERRY

324 IST Building
School of Information Sciences and Technology
Penn State University
University Park, PA 16802

home:
229 Montauk Circle
Port Matilda, PA 16870
(814) 867-2368

Education: BS (Nuclear Engineering) 1973 Penn State University
MS (Nuclear Engineering) 1974 University of Michigan
PhD (Information Sciences and Tech.) 2005 Penn State University

Penn State University. Research Assistant in Information Sciences and Technology, 2002-present.

I am a Ph.D. student in the School of Information Sciences and Technology. My research interests include machine learning, inference, information fusion and decision-making in real world complex domains that are characterized by large amounts of uncertainty. My research is focused on the use of probabilistic graphical models (PGMs) such as Bayesian networks, influence diagrams, and Markov decision processes. Specifically, I am interested in tractable inference methods that can be applied to PGMs used to represent large distributed sensor networks operating in an uncertain dynamic environment that are robust in the face of sensor and communication noise, node failures, asynchronous and limited bandwidth communications, and limited energy supplies.

Publications: